EXHIBIT 1

WSGR Wilson Sonsini Goodrich & Rosati
PROFESSIONAL CORPORATION

650 Page Mill Road
Palo Alto, CA 94304-1050
PHONE 650.493.9300
FAX 650.493.6811
www.wsgr.com

March 6, 2007

VIA ELECTRONIC MAIL

James Pooley, Esq.
POOLEY & OLIVER, LLP
Five Palo Alto Square
3000 El Camino Real
Palo Alto, CA 94306

Re: ***Synopsys v. Magma Design Automation***

Dear Jim:

As you know, Synopsys intends to amend its reply to assert inequitable conduct defenses with respect to the '116 and '093 patents. Per your request to Ron Shulman, our proposed amended reply is enclosed. A redline version, showing the changes made, is also enclosed. Please let us know by close of business Thursday whether Magma will stipulate to the proposed amendment.

In recent weeks, both parties have dropped various claims and defenses, and these changes are reflected in our proposed amended reply. In particular, we have deleted paragraphs 164-233 from our proposed reply. In addition, we assume that Magma has withdrawn the allegations in paragraphs 1-8, 30-36, 37-42, 50-51, 63-104, 105-119, 135-136, 141-163, and 172-175 of Magma's second amended answer, as these allegations relate solely to claims and defenses that are no longer asserted. We have therefore removed the paragraphs responding to these allegations from our proposed reply. If our assumption is incorrect, please let us know by close of business Thursday which paragraphs Magma has not withdrawn.

Thank you, and please feel free to call me if you have any questions.

Very truly yours,

WILSON SONSINI GOODRICH & ROSATI
Professional Corporation

[illegible]

Julie M. Holloway

C:\NrPortbl\PALIB1\ANB\3069847_1.DOC

EXHIBIT 2

Subject: Synopsys v. Magma Design Automation

From: James Pooley [mailto:jpooley@pooleyoliver.com]
Sent: Thursday, March 08, 2007 3:34 PM
To: Holloway, Julie
Cc: Scott Oliver
Subject: Re: Synopsys v. Magma Design Automation

Julie: we agree that your proposed amended pleading is correct as to the portions you have stricken, and we agree that the paragraphs of Magma's counterclaim you reference are allegations which it will not pursue. However, we will not stipulate to the two new claims of inequitable conduct/unenforceability. Thanks. Jim

----- Original Message -----
From: Holloway, Julie <JHolloway@wsgr.com>
To: James Pooley
Sent: Wed Mar 07 16:12:25 2007
Subject: FW: Synopsys v. Magma Design Automation

Jim,

We can certainly wait until the end of the day Thursday. We'll copy Scott on future communications. Thanks -
Julie M. Holloway
Wilson Sonsini Goodrich & Rosati
650 Page Mill Road
Palo Alto, CA 94304
(650) 320-4562

From: Baranski, Adrienne
Sent: Wednesday, March 07, 2007 4:09 PM
To: Holloway, Julie
Subject: FW: Synopsys v. Magma Design Automation

I believe this was meant for you but sent to me.

From: James Pooley [mailto:jpooley@pooleyoliver.com]
Sent: Wednesday, March 07, 2007 4:10 PM
To: Baranski, Adrienne
Cc: Scott Oliver
Subject: RE: Synopsys v. Magma Design Automation

Julie: thank you for your letter. Unfortunately, I was out of the office and unable to read it right away on my blackberry, and because the email didn't say what it was about, it wasn't forwarded right away to someone else. I'll get back to you as quickly as I can with a substantive answer, but it might not be until tomorrow. In the future, on anything as important as this, please copy Scott Oliver, and also include a title and cover message in your email that alerts us to what you're sending. Thanks.
Jim

From: Baranski, Adrienne [mailto:ABaranski@wsgr.com]
Sent: Tuesday, March 06, 2007 2:45 PM
To: James Pooley
Subject: Synopsys v. Magma Design Automation

Please see attached.

<<Letter.pdf>> <<AmendedReply.pdf>> <<Redline.pdf>>

Adrienne Baranski
Assistant to James Yoon, Julie Holloway,
  Kelley Moohr & Susan Bower
WILSON SONSINI GOODRICH & ROSATI
650 Page Mill Road
Palo Alto, CA 94304
Tel.: (650) 845-5957
Fax: (650) 565-5100
abaranski@wsgr.com

This email and any attachments thereto may contain private, confidential, and privileged material for the sole use of the intended recipient. Any review, copying, or distribution of this email (or any attachments thereto) by others is strictly prohibited. If you are not the intended recipient, please contact the sender immediately and permanently delete the original and any copies of this email and any attachments thereto.

EXHIBIT 3

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

SYNOPSYS, INC.,	)	
a Delaware corporation,	)	
	)	
Plaintiff,	)	
	)	
v.	)	C.A. No. 05-701 (GMS)
	)	
MAGMA DESIGN AUTOMATION,	)	
a Delaware corporation,	)	
	)	
Defendant.	)	

***SECOND* AMENDED FINAL JOINT CLAIM CONSTRUCTION CHARTS**

MORRIS, NICHOLS, ARSHT & TUNNELL LLP
Jack B. Blumenfeld (#1014)
Karen Jacobs Louden (#2881)
Leslie A. Polizoti (#4299)
1201 N. Market Street
P.O. Box 1347
Wilmington, DE 19899-1347
(302) 658-9200
Attorneys for Plaintiff Synopsys, Inc.

FISH & RICHARDSON P.C.
William J. Marsden, Jr. (#2247)
Kyle Wagner Compton (#4693)
919 N. Market Street
P.O. Box 1114
Wilmington, DE 19899
(302) 652-5070
Attorneys for Defendant Magma Design Automation, Inc.

November 28, 2006

The parties have met and conferred, and present the following ***Second*** Amended Joint Claim Construction Charts.

I. STIPULATED CONSTRUCTIONS

During the meet and confer process, the parties have agreed upon the constructions set forth in **Exhibit A** for U.S. Patent No. 6,192,508 ("the '508 Patent"), U.S. Patent No. 6,505,328 ("the '328 Patent"), U.S. Patent No. 6,519,745 ("the '745 Patent"), U.S. Patent No. 6,857,116 ("the '116 Patent"), and U.S. Patent No. 6,854,093 ("the '093 Patent").

II. CLAIM TERMS REQUIRING CONSTRUCTION BY THE COURT

The parties' ***Second*** Amended Joint Claim Construction Charts for the '508 Patent, the '745 Patent, and the '116 Patent are attached hereto as **Exhibits B, C,** and **D,** respectively. Each chart identifies the disputed claim terms in each patent, each party's proposed construction for the disputed claim terms, and each party's identification of the intrinsic and/or extrinsic evidence supporting its proposed construction.

This Second Amended Joint Claim Construction Chart differs from the Amended Chart submitted on November 3, 2006 in the following respects:

- Exhibit A has been modified to reflect the parties' agreement on the construction of the terms "limits" from the '508 Patent and "buckets" from the '745 Patent, and Exhibits B and C (pertaining to disputed terms) have been modified to delete these terms.
- The parties have modified the proposed constructions for "reducing constraints on a subsequent placement step" in the '508 Patent (see Exhibit B).

- There have been no changes to Exhibit D.

The parties jointly and respectfully request that, if the Court deems it appropriate, the Court include the list of agreed upon claim term constructions attached as Exhibit A in the ultimate claim construction order. In the alternative, the parties agree that this list of agreed-upon claim terms and constructions be considered a binding stipulation between the parties.

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ Leslie A. Polizoti
Jack B. Blumenfeld (#1014)
Karen Jacobs Louden (#2881)
Leslie A. Polizoti (#4299)
1201 N. Market Street
P.O. Box 1347
Wilmington, DE 19899-1347
(302) 658-9200
Attorneys for Plaintiff Synopsys, Inc.

FISH & RICHARDSON P.C.

/s/ Kyle Wagner Compton
William J. Marsden, Jr. (#2247)
Kyle Wagner Compton (#4693)
919 N. Market Street
P.O. Box 1114
Wilmington, DE 19899
(302) 652-5070
Attorneys for Defendant Magma Design Automation, Inc.

OF COUNSEL:

James J. Elacqua
Valerie M. Wagner
DECHERT LLP
1117 California Avenue
Palo Alto, CA 94304
(650) 813-4848

George G. Gordon
DECHERT LLP
Cira Center
2929 Arch Street
Philadelphia, PA 19104-2808
(215) 261-2382

Rebecca P. Dick
DECHERT LLP
1775 Eye Street, N.W.
Washington, DC 20006-2401
(202) 261-3432

OF COUNSEL:

FISH & RICHARDSON P.C.
Katharine Kelley Lutton
Tamara Fraizer
500 Arguello Street, Suite 500
Redwood City, CA 94063
(650) 839-5070

James Pooley
L. Scott Oliver
POOLEY & OLIVER LLP
Five Palo Alto Square, 7th Floor
Palo Alto, CA 94306-2121
(650) 739-7020

November 28, 2006

547164

EXHIBIT A

EXHIBIT A

Synopsys, Inc. v. Magma Design Automation, Inc.
(Case No. 05-701-GMS)

LIST OF AGREED UPON CLAIM TERM CONSTRUCTIONS

'508 Claim Term	**Agreed Construction**
fanins (Claim 11)	inputs to a circuit.
fanouts (Claims 9, 10)	terminals connected to the output of a gate.
gate (Claims 7-11)	a device having an output and one or more inputs, wherein the output is determined by the input, also referred to as a "cell."
initial placement (Claims 1-18)	a first placement of the integrated circuit elements of an integrated circuit, which can then be modified.
limits (Claims 1-18)	upper bounds.
logically equivalent (Claim 7)	performing the same logical function.
logic modification (Claims 1-18)	a modification of the actual logic of the circuit (as opposed to mere repositioning or trading places between gates).
modifying logic (Claims 4, 5, 7-11)	modifying the actual logic of the circuit (as opposed to mere repositioning or trading places between gates).

'508 Claim Term	**Agreed Construction**
net (Claims 12-14, 16, 18)	a connection between integrated circuit elements.
netlist (Claims 12-14, 16, 18)	a description of the connections between integrated circuit elements.
pin (Claim 8)	an input or an output of a gate. [Construction for the '508 and '328 patents only]
placement (Claims 1-18)	assigning the cells of the circuit to locations on the chip.
timing slack (Claim 5)	the degree to which a timing requirement is met in an integrated circuit design.
means for, subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design to allow congestion of the placement to be improved (Claim 17)	Construction of a portion of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is "performing logic modifications within selected bins of the integrated circuit design." The corresponding structure includes a computer executing algorithms for performing logic modifications within selected bins of the integrated circuit design, each logic modification including one of: 1. fanout splitting using buffering as shown in Figures 3A and 3B (*see* specification at col. 4, lines 23-45 and Figures 3A and 3B). In other words, adding buffers at the output pin of a gate and distributing the fanouts of the gate between the added buffers. 2. fanout splitting using node splitting as shown in Figures 3A and 3C (*see* specification at col. 4, lines 23-36 and 46-52, and Figures 3A and 3C). In other

'508 Claim Term	Agreed Construction
	words, replacing a gate with at least two copies of the gate, each of the copies fanning out to some of the fanouts of the gate. 3. intra-bin pin density logic optimization as shown in Figures 4A and 4B (*see* specification at col. 5, lines 1-12, and Figures 4A and 4B). In other words, replacing a set of gates in a bin with a different but logically equivalent set that has fewer pins. 4. input-splitting logic optimization as shown in Figures 4A, 4B, 6A and 6B (*see* specification at col. 5, lines 26-44, and Figures 4A, 4B, 6A and 6B). In other words, replacing a gate having a plurality of input pins with a set of gates, each one of which has fewer input pins. 5. Inter-bin pin density logic optimization as shown in Figures 5A and 5B (*see* specification at col. 5, lines 13-25, and Figures 5A and 5B). In other words, moving connections between gates and across bins to reduce pin density in a congested bin; and 6. performing logic modifications that speed up part of the circuit to improve timing slack in that part of the circuit, each logic modification including remapping or buffering (*see* Specification at col. 3, lines 65 through col. 4, line 7).
means, subject to limits on the increase in area of integrated circuit elements within a bin, performing logic modifications within selected bins of the integrated circuit design; wherein the logic modifications improve timing of selected nets belonging to the selected bins, reducing constraints on a subsequent placement step (Claim 18)	Construction of a portion of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is "performing logic modifications within selected bins of the integrated circuit design; wherein the logic modifications improve timing of selected nets belonging to the selected bins." The corresponding structure is a computer executing algorithms for: Performing logic modifications that speed up

'508 Claim Term	Agreed Construction
	part of the circuit to improve timing slack in that part of the circuit, each logic modification including either remapping or buffering. *See* Specification at col. 3, lines 65 through col. 4, line 7.

'328 Claim Term	Agreed Construction
active memory (Claims 11-13)	temporary data storage that can be read and changed while the computer is in use.
adapted (Claims 1-17)	suited.
area query (Claims 1-17)	a request for objects intersecting a specified area (synonymous with region query).
associated (Claims 1-17)	having a relationship with.
common data model (Claims 1-17)	a shared data model that does not require translation between the design tools.
data representation (Claims 1-17)	data objects, at least some of which stand for elements in an integrated circuit.
disk storage (Claim 13)	persistent storage by means of a disk.
logically correlated (Claims 1-17)	having a logical relationship.

'328 Claim Term	**Agreed Construction**
maintained (Claims 11, 12, 13)	kept.
net (Claim 9)	a connection between integrated circuit elements.
netlist (Claims 1-17)	a description of the connections between integrated circuit elements.
pin (Claim 10)	an input or an output of a gate. [Construction for the '508 and '328 patents only]
placement (Claim 15)	assigning the cells of the circuit to locations on the chip.

'745 Claim Term	**Agreed Construction**
bucket (Claims 1-8)	A rectangular, coarse placement region within the chip's core area. * "Placement region" means "a region on the chip's core defined before or during placement, into which cells have been or are later placed."
maintaining a congestion score for each bucket (Claims 1-8)	keeping or keeping up an adjusted congestion score during routing.
a range of congestion scores is equivalent to a given spacing configuration for wires in a bucket (Claim 6)	a given range of congestion scores corresponds to a particular spacing between wires in a bucket.

'745 Claim Term	Agreed Construction
when routing a wire through a bucket, modifying the congestion score accordingly (Claims 1-8)	every time a wire is routed through a bucket, modifying the congestion score accordingly.
routing (Claims 1-8)	interconnecting the components of the circuit with wiring.
lateral capacitance (Claim 8)	capacitance due to the overlap along the side walls of a wire with adjacent signal wires.

'116 Claim Term	Agreed Construction
netlist (Claims 1-52)	a description of the connections between integrated circuit elements.
prior integrated circuit (Claims 1-52)	an integrated circuit that has undergone the physical design phase.
pin (Claims 2, 10, 13, 14, 16, 24, 27-52)	a location at the edge of a block where a signal can enter the block or exit the block. [Construction for the '116 patent only]
pin assignment (Claims 2, 13, 14, 16, 27-52)	assignment of pin location.
using said netlist and said physical design information (Claims 8, 9, 22, 23, 29-52)	using the netlist and the physical design information for the purpose of improving the current integrated circuit
abutted-pin (Claims 10, 24, 38, 50)	pin(s) physically touching the edge or boundary of each block and resting against the edge or boundary of another block such that the pin(s) of one block abut(s) the pin(s) of another block.
hierarchical physical design (Claims 10, 24, 38, 50)	a physical design with two or more levels.
obstruction (Claims 11, 12, 25, 26, 36, 37, 48, 49)	an object or region in which further placement or routing is impeded
placement (Claims 13, 14, 27-52)	assigning the cells of the circuit to locations on the chip.
port (Claims 13, 14, 27-52)	an input or output internal to a block that may have connections to other ports in other blocks.

'116 Claim Term	Agreed Construction
to determine pin assignments (Claims 13, 14, 27-52)	to determine pin assignments for the current integrated circuit.
based on said top-level route (Claims 29-52)	using the top-level route for the purpose of improving pin assignments in the current integrated circuit
pressing (Claims 14, 28)	Removing the top-level objects within the boundary of a block from the top-level netlist and merging those objects into the block-level netlist of that block.

'093 Claim Term	Agreed Construction
pressing (Claims 1-40)	Removing the top-level objects within the boundary of a block from the top-level netlist and merging those objects into the block-level netlist of that block.
netlist (Claims 1-40)	a description of the connections between integrated circuit elements.
hierarchical state (Claims 1-40)	characterized by having at least two levels.
abutted-pin (Claims 2-4, 12-14, 22-24, 32-34)	pin(s) physically touching the edge or boundary of each block and resting against the edge or boundary of another block such that the pin(s) of one block abut(s) the pin(s) of another block.
hierarchical physical design (Claims 2-4, 12-14, 22-24, 32-34)	a physical design with two or more levels.
placement (Claims 21-40)	assigning the cells of the circuit to locations on the chip.

'093 Claim Term	**Agreed Construction**
press property (Claims 21-40)	a property of the top-level object that is stored, such that, if the property is present, the portion of the top-level object within the boundary of the block retains its location when pressed into the block, and, if the property is not present, the portion of the top-level object generally does not retain its location when pressed into the block.

'733 Claim Term	**Agreed Construction**
HDL (Claims 1-14)	abbreviation for "Hardware Description Language" – a computer language for a high-level description of an integrated circuit design
Netlist (Claims 1-26)	a description of the connections between integrated circuit elements
Partitioning information (Claims 1-26)	data representative of the sets of re-orderable scan cells
Clock domain (Claims 2, 9, 16, and 22-26)	a region of a circuit in which the timing behavior is identical or very similar
Edge sensitivity types (Claims 3, 10, 17, 22- 26)	the type of sensitivity of a cell (e.g., a rising edge sensitivity or a falling edge sensitivity or a positive edge or negative edge)
Reconfigurable multiplexer (Claims 4, 11, 18, and 24)	a switch used in a scan chain
Clock skew tolerance levels (Claims 5, 12 and 19)	the difference in time allowed between the arrival of the clock signals at two or more places (e.g., within an acceptable time window).
Surrounding cone logic (Claims 6, 13, 20, and 25)	a group of cells feeding a particular cell or being fed by a cell
Output switching times (Claims 7, 14, and 21)	the time at which a cell's output transitions to a given state
Scan chain (Claims 1-26)	scan cells connected together to form a chain or sequence

'733 Claim Term	Agreed Construction
Scan cells (Claims 1-26)	special memory cells specifically designed for test, which can be scanned

'501 Claim Term	Agreed Construction
Netlist (Claims 1-26)	a description of the connections between integrated circuit elements
Partitioning information (Claims 1-26)	data representative of the sets of re-orderable scan cells
Clock domain (Claims 2, 9, 16, 25, and 26)	a region of a circuit in which the timing behavior is identical or very similar
Edge sensitivity types (Claims 3, 10, 17, 25, and 26)	the type of sensitivity of a cell (e.g., a rising edge sensitivity or a falling edge sensitivity or a positive edge or negative edge)
Reconfigurable multiplexer (Claims 4, 11, 18, and 26)	a switch used in a scan chain
Clock skew tolerance levels (Claims 5, 12, 19, and 26)	the difference in time allowed between the arrival of the clock signals at two or more places (e.g., within an acceptable time window).
Surrounding cone logic (Claims 6, 13, 20, and 26)	a group of cells feeding a particular cell or being fed by a cell
Output switching times (Claims 7, 14, and 21)	the time at which a cell's output transitions to a given state
Scan chain (Claims 1-26)	scan cells connected together to form a chain or sequence
Scan cells (Claims 1-26)	special memory cells specifically designed for test, which can be scanned
Simultaneously switching output requirements (Claim 26)	limits to the number of output pins that can switch at one time

EXHIBIT B

EXHIBIT B

<u>Synopsys, Inc. v. Magma Design Automation, Inc.</u>
(Case No. 05-701-GMS)

U.S. Patent No. 6,192,508

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
bins (Claims 1-18)	one or more regions.	more than one bin.	**INTRINSIC EVIDENCE** Claim 1 preamble. (Col. 6, ll. 46-49). "It is important that the area used by the logic optimizations be monitored. Because the current placement (at the time of the logic optimizations) is based on a certain area of all the bins, if this information changes, then the placement may no longer be appropriate. The change may result in placement being done again at each step, and possibly the process never converging. Monitoring of the area used in order to preserve the feasibility of the placement is done by placing an upper bound on the area of each bin. The proposed logic optimizations are only allowed to increase the bin area to the upper bound. Bounding the increase in bin area guarantees convergence of the placement process." (Col. 5, ll. 58 – col. 6, ll. 2). "By the present amendment, the claims would be amended to account for the possibility of performing the present invention using only a single bin (i.e.,	**INTRINSIC EVIDENCE** 2:21-6:44 Figure 4a Claims 1-18 Amendment Under 37 C.F.R. § 312, dated March 6, 2000 (SYN1499155-158) in the '508 Patent Prosecution History (SYN1498962-SYN1499212). **EXTRINSIC EVIDENCE** Magma reserves the right to present expert opinion testimony by written declaration.

<u>**Synopsys, Inc. v. Magma Design Automation, Inc.**</u>
(Case No. 05-701-GMS)

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			one encompassing the entire integrated circuit) as opposed to multiple bins. This change is not believed to affect patentability of the claims. Entry of the amendment is respectfully requested." (SYN1499158). Amendment Under 37 C.F.R. § 312, dated March 6, 2000 (SYN1499155-158) in the '508 Patent Prosecution History (SYN1498962-SYN1499212). Response to Rule 312 Communication, dated May 9, 2000 (SYN1499159), in the '508 Patent Prosecution History (SYN1498962-SYN1499212). **EXTRINSIC EVIDENCE** "**region** *n. Abbr.* **reg. 1**. Any large, usually continuous segment of a surface or space; an area." *The American Heritage Dictionary of the English Language*, p. 1095 (William Morris, ed., Houghton Mifflin Co., 1976). Synopsys reserves the right to present expert opinion testimony by written declaration.	

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
bin (Claims 1-15, 17, 18)	a region.	a rectangular (or square) portion of an integrated circuit bounded by gridlines.	Same evidence as cited for "**bins**."	**INTRINSIC EVIDENCE** 2:21-6:44 Claim 1 **EXTRINSIC EVIDENCE** "Application-Specific Integrated Circuits," Michael John Sebastian Smith, 1997, p. 882-885. Chang, *et.al.*, "Physical Hierarchy Generation with Routing Congestion Control," ISPD 2002. U.S. Patent No. 5,847,965. U.S. Patent No. 6,442,743. Magma reserves the right to present expert opinion testimony by written declaration.
in an attempt to improve congestion by taking advantage of the logic modifications (Claims 2-11, 13, 14)	to relieve congestion where opportunities are provided by logic modifications.	with the purpose of reducing congestion by taking advantage of more than one logic modification.	**INTRINSIC EVIDENCE** "The second method involves modifying the topology of the circuit by adding gates while maintaining the functionality, such that the added gates can then be moved by the placement steps to relieve congestion.	*See below* at "to allow congestion of the placement to be improved."

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			"An important aspect of the optimizations, specifically directed towards helping placement relieve congestion, is the ability to undo modifications if placement does not actually use the modifications. The undo capability ensures that no area/power resources are wasted for transformations the are not used as intended." (Col. 2, 43-52). "Modification of logic to potentially improve circuit congestion (Step 5). Techniques such as fanout splitting are used for this. "Placement modification to take advantage of the preceding modifications (Step 6)." (Col. 4, ll. 8-12). "For many of the congestion relieving logic synthesis methods proposed as part of placement, there are two important issues that this invention addresses. In most cases, logic synthesis cannot itself improve congestion, but rather only provide opportunities for placement to improve congestion, it is important to track which of these opportunities are	

'508 Patent Claims	**Synopsys' Proposed Construction**	**Magma's Proposed Construction**	**Synopsys' Support**	**Magma's Support**
			actually used. Any unused opportunities may result in wasted resources, since the logic optimization step used to create them typically uses additional area and power (for faster cells) or additional gates. The use of the logic optimizations during placement is therefore actively tracked. Any unused optimizations are undone to ensure that there are no wasted resources." (Col. 5, ll. 45-57). **EXTRINSIC EVIDENCE** Synopsys reserves the right to present expert opinion testimony by written declaration.	
means for calculating congestion of the initial placement (Claims 17, 18)	Construction of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is "calculating congestion of the initial placement." The corresponding structure is a computer executing algorithms for: • calculating congestion for the initial placement using	Construction of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is calculating congestion of the initial placement. The corresponding structure is a computer executing algorithms for: calculating the total number of pins in the bin divided by the	**INTRINSIC EVIDENCE** Corresponding structure is found in the specification at: Figure 7 Col. 3, ll. 35-38 Col. 4, ll. 61-67 Col. 6, ll. 3-23 "Interconnection models for interconnects between bins and within bins provide both delay estimates for each interconnect in the circuit, as well as congestion estimates for each bin in	**INTRINSIC EVIDENCE** Fig. 7 4:61-63 6:3-23 **EXTRINSIC EVIDENCE** Magma reserves the right to present expert opinion testimony by written declaration.

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
	interconnection models for interconnects between bins or within bins (Col. 3, ll. 35-38); or • calculating congestion for the initial placement in accordance with an algorithm that calculates the total number of pins in the bin divided by the total routable area in the bin (Col. 4, ll. 61-67).	total routable area in the bin.	the circuit." (Col. 3, ll. 35-38). "One measure of the congestion in a bin is given by pin density, calculated as the total number of pins in the bin divided by the total routable area in the bin. Here a pin refers to either an input of an output of a cell. It is desirable to get a lower congestion since that is likely to make routing easier. It is possible for logic optimizations to directly reduce this measure of congestion." (Col. 4, ll. 61-67). "Referring to FIG. 7, a diagram is shown of a computer system that may be used to practice the present invention." (Col. 6, ll. 12-13). **EXTRINSIC EVIDENCE** Synopsys reserves the right to present expert opinion testimony by written declaration.	
means for performing an initial placement of integrated circuit elements within bins on the design layout	Construction of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is "performing an initial placement	Construction of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is performing an initial placement	**INTRINSIC EVIDENCE** Corresponding structure is found in the specification at: Figure 7 Col. 3, ll. 31-35	**INTRINSIC EVIDENCE** None. **EXTRINSIC EVIDENCE**

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
(Claim 17)	of integrated circuit elements within bins on the design layout." The corresponding structure is: • an electronic design automation placement tool; • a computer executing an algorithm for placing cells in one or more regions using a placement tool that partitions cells into one or more regions at each stage of the placement (Col. 3, ll. 31-35); and • a computer executing an algorithm for placing cells in accordance with a placement algorithm that is limited by the topology of the circuit (Col. 4, ll. 23-29).	of integrated circuit elements within bins on the design layout. The corresponding structure is: [No corresponding structure is disclosed.]	Col. 4, ll. 23-29 Col. 6, ll. 3-23 "The present invention may be used in conjunction with an electronic design automation placement tool. In accordance with an exemplary embodiment of one such placement tool, at each stage in cell placement, the cells are partitioned into a number of bins." (Col. 3, ll. 31-35). "Placement algorithms are limited in how they can place cells by the topology of the circuit. If the output of cell A is connected to (also referred to as 'fanning out to') four different terminals in different cells (indicated by the numbers 1-4) in FIG. 3(a), then the placement of A is strongly influenced by the placement of cells corresponding to these terminals." (Col. 4, ll. 23-29). "Referring to FIG. 7, a diagram is shown of a computer system that may be used to practice the present invention." (Col. 6, ll. 12-13). **EXTRINSIC EVIDENCE**	Magma reserves the right to present expert opinion testimony by written declaration.

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			Synopsys reserves the right to present expert opinion testimony by written declaration. Application-Specific Integrated Circuits, Michael John Sebastion Smith, © 1997 by Addison Wesley Longman, Inc., pp. 873-893.	
means for performing an initial placement of integrated circuit elements within bins on the design layout (Claim 18)	Construction of a portion of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is "performing an initial placement of integrated circuit elements within bins on the design layout." The corresponding structure is: • an electronic design automation placement tool; • a computer executing an algorithm for placing cells in one or more regions using a placement tool that partitions cells into one or more regions at each stage of the placement (Col. 3, ll. 31-35); and	Construction of a portion of this phrase is governed by 35 U.S.C. § 112, ¶ 6. The claimed function is performing an initial placement of integrated circuit elements within bins on the design layout. The corresponding structure is: [No corresponding structure is disclosed.]	**INTRINSIC EVIDENCE** Corresponding structure is found in the specification at: Figure 7 Col. 3, ll. 31-35 Col. 4, ll. 23-29 Col. 6, ll. 3-23 "The present invention may be used in conjunction with an electronic design automation placement tool. In accordance with an exemplary embodiment of one such placement tool, at each stage in cell placement, the cells are partitioned into a number of bins." (Col. 3, ll. 31-35). "Placement algorithms are limited in how they can place cells by the topology of the circuit. If the output of cell A is connected to (also referred to as 'fanning	*See above* at "means for performing an initial placement of integrated circuit elements within bins on the design layout."

Synopsys, Inc. v. Magma Design Automation, Inc.
(Case No. 05-701-GMS)

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
	• a computer executing an algorithm for placing cells in accordance with a placement algorithm that is limited by the topology of the circuit (Col. 4, ll. 23-29).		out to') four different terminals in different cells (indicated by the numbers 1-4) in FIG. 3(a), then the placement of A is strongly influenced by the placement of cells corresponding to these terminals." (Col. 4, ll. 23-29). "Referring to FIG. 7, a diagram is shown of a computer system that may be used to practice the present invention." (Col. 6, ll. 12-13). **EXTRINSIC EVIDENCE** Synopsys reserves the right to present expert opinion testimony by written declaration. Application-Specific Integrated Circuits, Michael John Sebastion Smith, © 1997 by Addison Wesley Longman, Inc., pp. 873-893.	
reducing constraints on a subsequent placement step (Claims 12-14, 16, 18)	reducing one or more constraints on a subsequent placement step.	Plain meaning	**INTRINSIC EVIDENCE** "The circuit has timing constraints imposed on it that it needs to satisfy. The delay estimates of the interconnection, combined with the delays of the cells and the timing constraints imposed on the design, are	

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			converted to timing slack information for each part of the circuit. A negative timing slack indicates that that part of the circuit is not meeting the timing constraints. A positive slack indicates that that part of the circuit is producing its result faster that is needed and can thus be slowed down without violating its timing constraints. More generally, 'slack' is defined herein as a measure of the degree to which a timing requirement is met in an integrated circuit design. "The traditional role of logic synthesis has been to identify areas of the circuit which have negative timing slack and then modify the circuit so as to fix this problem. As described herein, logic synthesis is used to aid placement to achieve both acceptable delays and congestion, by making circuit modifications that increase the timing slack in the congested parts." (Col. 3, ll. 38-56). **EXTRINSIC EVIDENCE** Synopsys reserves the right to present expert opinion testimony by written declaration.	

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			See also the same evidence as cited for "**bins.**"	
selected bins (Claims 1-18)	one or more selected regions.	more than one bin selected based on congestion.	Same evidence as cited for "**bins.**"	**INTRINSIC EVIDENCE** 2:28-33 2:37-43 3:50-56 3:61-62 4:22-23 4:53-55 4:59-60 4:64-67 5:13-15 6:30-35 Fig. 2 Claims 1, 2, 3, 4, 17, and 18. Applicant Response, 1/20/2000, at 2, 3, 4. Notice of Allowability, 1/31/2000, at 2. **EXTRINSIC EVIDENCE** Magma reserves the right to present expert opinion testimony by written declaration.
to allow congestion of the placement to be	to provide opportunities for placement to improve congestion.	with the purpose of reducing congestion of the placement.	**INTRINSIC EVIDENCE** "The second method involves modifying	**INTRINSIC EVIDENCE** 2:21-22

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
improved (Claims 1-11, 15, 17)			the topology of the circuit by adding gates while maintaining the functionality, such that the added gates can then be moved by the placement steps to relieve congestion. "An important aspect of the optimizations, specifically directed towards helping placement relieve congestion, is the ability to undo modifications if placement does not actually use the modifications. The undo capability ensures that no area/power resources are wasted for transformations the are not used as intended." (Col. 2, 43-52). "Modification of logic to potentially improve circuit congestion (Step 5). Techniques such as fanout splitting are used for this. "Placement modification to take advantage of the preceding modifications (Step 6)." (Col. 4, ll. 8-12). "For many of the congestion relieving logic synthesis methods proposed as part of placement, there are two important	2:28-33 2:47-52 3:50-56 3:65-4:7 4:13-14 4:22-23 4:59-60 4:64-67 5:45-57 6:30-35. Applicant Response, 1/20/2000, at 2, 3, 4. Notice of Allowability, 1/31/2000, at 2. **EXTRINSIC EVIDENCE** U.S. Patent No. 6,099,580. "Application-Specific Integrated Circuits," Michael John Sebastian Smith, 1997, ch. 16. "Physical Design CAD in Deep Sub-Micron Era," Mitsuhashi *et al.*, 1996, EURO-DAC '96. "Fanout-tree Restructuring Algorithm for Post-placement Timing Optimization," T. Aoki, 1995.

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			issues that this invention addresses. In most cases, logic synthesis cannot itself improve congestion, but rather only provide opportunities for placement to improve congestion, it is important to track which of these opportunities are actually used. Any unused opportunities may result in wasted resources, since the logic optimization step used to create them typically uses additional area and power (for faster cells) or additional gates. The use of the logic optimizations during placement is therefore actively tracked. Any unused optimizations are undone to ensure that there are no wasted resources." (Col. 5, ll. 45-57). "This invention will significantly reduce, if not eliminate, the iterations needed by considering not only the impact of interconnect during logic optimization of area/timing, but also at the same time doing logic optimization to help placement relieve congestion and thus generate a circuit that is easily routable." (Col. 6, ll. 30-35). **EXTRINSIC EVIDENCE** "**allow** *tr.v.* … **6**. To provide (the	Magma reserves the right to present expert opinion testimony by written declaration.

'508 Patent Claims	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			needed amount): *allow funds in case of emergency*." *The American Heritage Dictionary of the English Language*, p. 35 (William Morris, ed., Houghton Mifflin Co., 1976). **EXTRINSIC EVIDENCE** Synopsys reserves the right to present expert opinion testimony by written declaration.	

EXHIBIT C

EXHIBIT C

<u>Synopsys, Inc. v. Magma Design Automation, Inc.</u>
(Case No. 05-701-GMS)

U.S. Patent No. 6,519,745

'745 Claim Terms	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
congestion score (Claims 1-8)	the ratio of routing resources used so far to the total routing resources available.	a ratio measure of routing resources.	**INTRINSIC EVIDENCE** "The congestion score for a bucket is defined as the ratio of the routing resources used so far to the total routing resources available in the bucket." '745 patent at 8:33-36. '745 patent at 3:63-4:3; 4:10-18; 8:17-9:29; Fig. 7; abstract. **EXTRINSIC EVIDENCE** Michael J.S. Smith, <u>Application-Specific Integrated Circuits</u> 859-861 (1997) U.S. Patent No. 6,618,846, at 5:53-57. Definition of "congest": "[t]o overfill or overcrowd." THE AMERICAN HERITAGE COLLEGE DICTIONARY 301 (4th ed. 2002). Synopsys reserves the right to present expert opinion testimony by written declaration.	**INTRINSIC EVIDENCE** 3:63-4:3 4:10-18 8:30-45 Fig. 7 Claim 2 **EXTRINSIC EVIDENCE** Expert opinion testimony by written declaration.

EXHIBIT D

Synopsys, Inc. v. Magma Design Automation, Inc.
(Case No. 05-701-GMS)

U.S. Patent No. 6,857,116

'116 Claim Terms	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
generating said physical design (Claims 1-28)	producing an improved physical design for the current integrated circuit	**Plain meaning – no construction needed.**	INTRINSIC EVIDENCE "Thus, the software tools of the physical design phase 910 can customize the current integrated circuit to avoid the problems of the prior integrated circuit and to realize the benefits of the prior integrated circuit." '116 patent at 9:16-19. "By using physical design information 930 (concerning the block-level of the prior integrated circuit) at the top-level of the current integrated circuit, the decisions made at the top-level with respect to the top-level objects of the current integrated circuit will be able to reduce the problems present in the prior integrated circuit and will be able to generate solutions to overcome the problems present in the prior integrated circuit, improving the optimization of the abutted-pin hierarchical physical design	**INTRINSIC EVIDENCE** 2:23-60 6:64-7:7 7:26-58 7:66-8:-65 8:66-9:43 Amendment and Response to Office Action of February 3, 2003 Office Action of April 28, 2003 **EXTRINSIC EVIDENCE** "Generate, v.: . . . 3. To bring about, give rise to, produce." *Oxford English Dictionary* (2d. Ed. 1989). Expert opinion testimony by written declaration.

'116 Claim Terms	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			process of the present invention. Thus, if the physical design information 930 has information about several prior integrated circuits, the current integrated circuit is more likely to be optimized." '116 patent at 9:23-34. "In sum, the pin assignments generated with the use of the physical design information of the prior integrated circuit (FIGS. 10A-10C) were more optimal than the pin assignments generated without the use of the physical design information of the prior integrated circuit (FIGS. 10A-10C)." '116 patent at 10:53-57. "a method of improving a physical design of a current integrated circuit…." Preamble of claims 1, 15. '116 patent at 2:39-43; 8:8-30; 8:45-46; 8:58-11:22.	

'116 Claim Terms	Synopsys' Proposed Construction	Magma's Proposed Construction	Synopsys' Support	Magma's Support
			Figs. 4-5, 8, 9B; 10A-C; 11A-C; 12A-C. Title of '116 patent. **EXTRINSIC EVIDENCE** Synopsys reserves the right to present expert opinion testimony by written declaration.	

EXHIBIT 4

US006505328B1

(12) **United States Patent**
Van Ginneken et al.

(10) **Patent No.: US 6,505,328 B1**
(45) **Date of Patent: Jan. 7, 2003**

(54) **METHOD FOR STORING MULTIPLE LEVELS OF DESIGN DATA IN A COMMON DATABASE**

(75) Inventors: **Lukas P. P. P. Van Ginneken**, San Jose, CA (US); **Patrick R. Groeneveld**, San Jose, CA (US); **Wilhelmus J. M. Philipsen**, Phoenix, AZ (US)

(73) Assignee: **Magma Design Automation, Inc.**, Cupertino, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/300,540**

(22) Filed: **Apr. 27, 1999**

(51) **Int. Cl.**⁷ **G06F 17/50**
(52) **U.S. Cl.** **716/7**; 716/8; 716/12
(58) **Field of Search** 716/1–21

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,187,668	A	*	2/1993	Okude et al.	716/8
5,212,650	A	*	5/1993	Hooper et al.	716/18
5,313,615	A	*	5/1994	Newman et al.	716/11
5,432,707	A	*	7/1995	Leung	716/2
5,452,226	A	*	9/1995	Hooper et al.	716/18
5,487,018	A	*	1/1996	Loos et al.	716/11
5,519,627	A	*	5/1996	Mahmood et al.	716/18
5,541,849	A	*	7/1996	Rostoker et al.	716/18
5,623,417	A	*	4/1997	Iwasaki et al.	716/18
5,666,288	A	*	9/1997	Jones et al.	716/17
5,696,693	A	*	12/1997	Aubel et al.	716/8
5,699,265	A	*	12/1997	Scepanovic et al.	716/10
5,726,902	A	*	3/1998	Mahmood et al.	716/6
5,727,187	A	*	3/1998	Lemche et al.	716/18
5,757,657	A		5/1998	Hathaway et al.	710/9
5,761,664	A		6/1998	Sayah et al.	707/100
5,764,534	A		6/1998	Goetting	716/11
5,818,729	A	*	10/1998	Wang et al.	716/9
5,841,663	A	*	11/1998	Sharma et al.	716/18
5,864,487	A	*	1/1999	Merryman et al.	716/6
5,956,497	A	*	9/1999	Ratzel et al.	716/1

(List continued on next page.)

OTHER PUBLICATIONS

Dutt ("Generic component library characterization for high level synthesis", Proceedings of the Fourth CSI/IEEE International Symposium on VLSI Design, 1991, Jan. 4, 1991, pp. 5–10).*

Dion, J. and Monier, L.M.: "Countour: A Title-based Gridless Router," *WRL Research Report 3/95*, Digital Western Research Laboratory.

Dion, J. and Monier, L.M.; "Recursive Layout Generation," *WRL Research Report 2/95*, Digital Western Research Laboratory.

Hwang., J., et al., "Generating layouts for self-emplementing modules"; Intl Workshop on Field Programmable Logic and Applications. FPGAS, GB, Abingdon, Aug. 31, 1998, pp. 525–529.

Singhal, A., et al., "Object oriented data modeling for VLSI/CAD," Proc. of the 8th Intl Conf. on VLSI Design, New Delhi, India, Jan. 4–7, 1995, pp. 25–29.

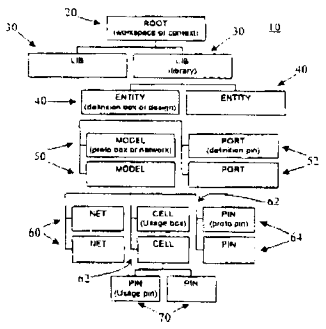
Fcanha, H.S.:, "Data astructtures for physical representation of VLSI." Software Eng'g Journal, GB, IEE, London, vol. 5, No. 6, Nov. 1, 1990.

Primary Examiner—Matthew Smith
Assistant Examiner—Phallaka Kik
(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop LLP

(57) **ABSTRACT**

An automated logic circuit design system uses a common database to store design data at different states of the design process, including data-flow graphs, netlists and layout descriptions. In this way, the need to translate circuit descriptions between tools is eliminated, thus leading to increased speed, flexibility and integration. The common database includes entities, models, cells, pins, busses and nets. The data-flow graphs are stored as graphs, the nodes in a graph as cells, and the edges as busses. Physical design data is available by storing the cells in a model in a KD tree. This allows queries on cells in the netlist located in the layout within arbitrary areas.

17 Claims, 4 Drawing Sheets



SYN0012146

U.S. PATENT DOCUMENTS

5,960,184	A	*	9/1999	Cleereman et al.	716/3
6,026,228	A	*	2/2000	Imai et al.	716/188
6,080,201	A	*	6/2000	Hojat et al.	703/14
6,145,117	A	*	11/2000	Eng	716/18
6,154,874	A	*	11/2000	Scepanovic et al.	716/13
6,216,258	B1	*	4/2001	Mohan et al.	716/17
6,263,483	B1	*	7/2001	Dupenloup	716/18
6,289,489	B1	*	9/2001	Bold et al.	716/1
6,308,309	B1	*	10/2001	Gan et al.	716/8

* cited by examiner

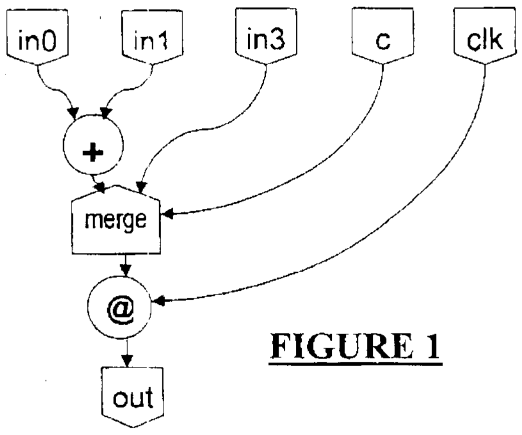


FIGURE 1

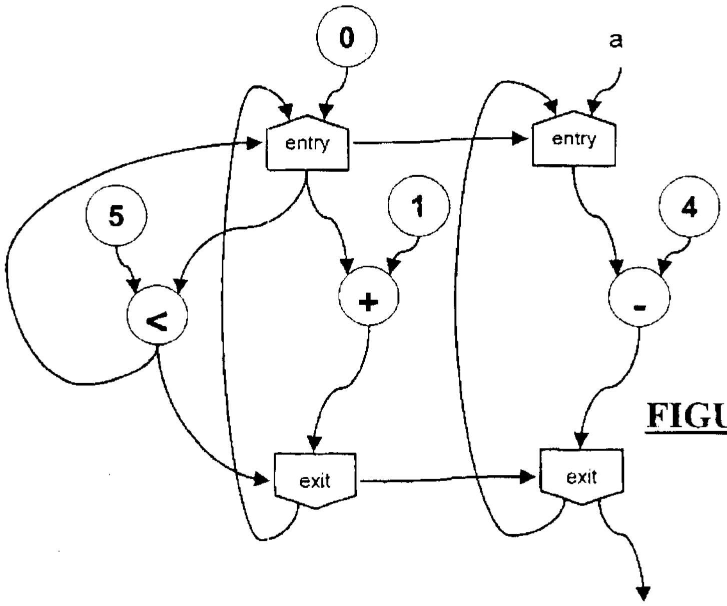


FIGURE 3

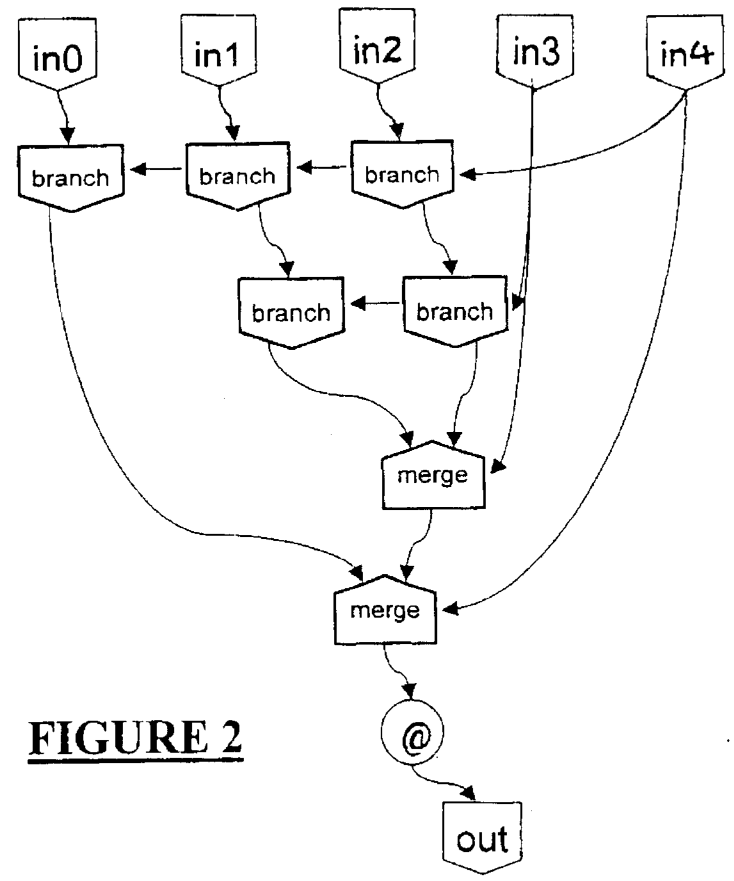

in0
in1
in2
in3
in4
branch
branch
branch
branch
branch
merge
merge
@
out

FIGURE 2

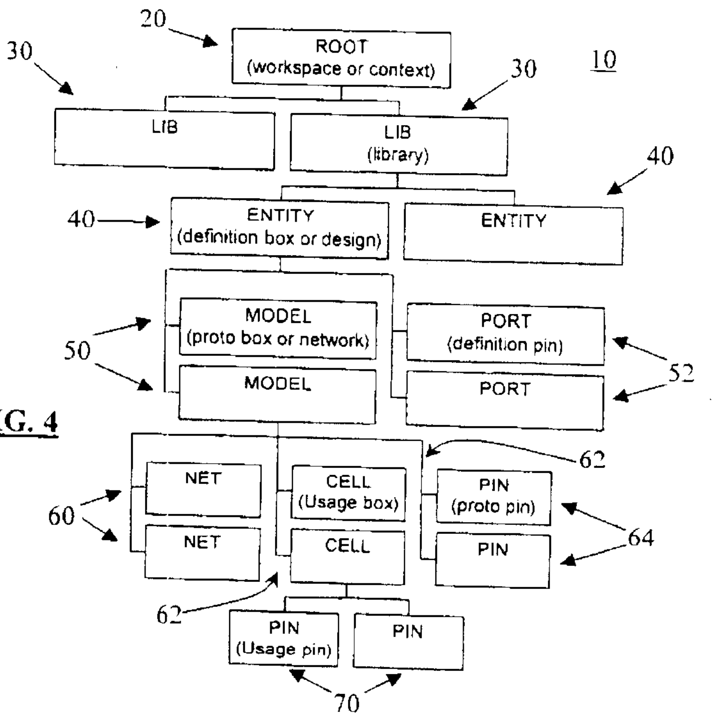


FIG. 4

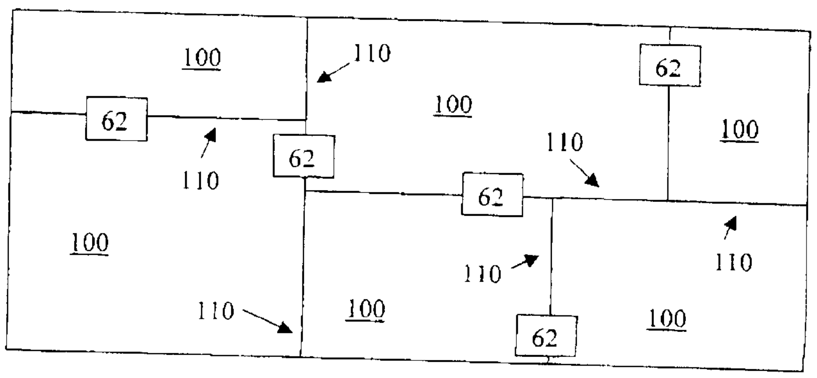


FIGURE 5

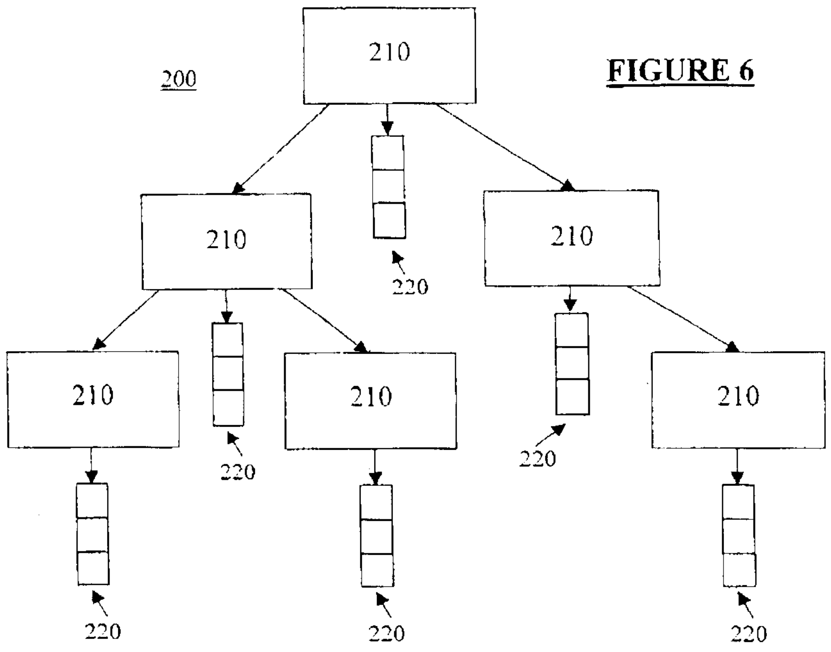


FIGURE 6

METHOD FOR STORING MULTIPLE LEVELS OF DESIGN DATA IN A COMMON DATABASE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is directed to digital logic design systems. More particularly, the invention is directed to automated digital logic synthesis and placement systems.

2. Background of the Related Art

Prior art computer aided design (CAD) systems for the design of integrated circuits and the like assist in the design thereof by providing a user with a set of software tools running on a computer. In the prior art, the process of designing an integrated circuit on a typical CAD system was done in several discrete steps using different software tools.

First, a schematic diagram of the integrated circuit is entered interactively to produce a digital representation of the integrated circuit elements and their interconnections. This representation may initially be in a hardware description language such as Verilog and then translated into a register transfer level (RTL) description in terms of pre-designed functional blocks, such as memories and registers. This may take the form of a data structure called a net list.

Next, a logic compiler receives the net list and, using a component database, puts all of the information necessary for layout, verification and simulation into object files whose formats are optimized specifically for those functions.

Afterwards, a logic verifier checks the schematic for design errors, such as multiple outputs connected together, overloaded signal paths, etc., and generates error indications if any such design problems exist. In many cases, the IC designer improperly connected or improperly placed a physical item within one or more cells. In this case, these errors are flagged to the IC designer so that the layout cells may be fixed so that the layout cells perform their proper logical operation. Also, the verification process checks the hand-laid-out cells to determine if a plurality of design rules have been observed. Design rules are provided to integrated circuit designers to ensure that a part can be manufactured with greater yield. Most design rules include hundreds of parameters and, for example, include pitch between metal lines, spacing between diffusion regions in the substrate, sizes of conductive regions to ensure proper contacting without electrical short circuiting, minimum widths of conductive regions, pad sizes, and the like. If a design rule violation is identified, this violation is flagged to the IC designer so that the IC designer can properly correct the cells so that the cells are in accordance with the design rules.

Then, using a simulator the user of the CAD system prepares a list of vectors representing real input values to be applied to the simulation model of the integrated circuit. This representation is translated into a form which is best suited to simulation. This representation of the integrated circuit is then operated upon by the simulator which produces numerical outputs analogous to the response of a real circuit with the same inputs applied. By viewing the simulation results, the user may then determine if the represented circuit will perform correctly when it is constructed. If not, he or she may re-edit the schematic of the integrated circuit, re-compile and re-simulate. This process is performed iteratively until the user is satisfied that the design of the integrated circuit is correct.

Then, the human IC designer presents as input to a logic synthesis tool a cell library and a behavioral model. The

behavioral circuit model is typically a file in memory which looks very similar to a computer program. The behavioral circuit model contains instructions which define logically the operation of the integrated circuit. The logic synthesis tool receives as input the instructions from the RTL circuit model (i.e., Verilog or VHDL) and the library cells from the library. The synthesis tool maps the instructions from the behavioral circuit model to one or more logic cells from the library to transform the behavioral circuit model to a gate schematic net list of interconnected cells. A gate schematic net list is a data base having interconnected logic cells which perform a logical function in accordance with the behavioral circuit model instructions. Once the gate schematic net list is formed, it is provided to a place and route tool.

The place and route tool is used to access the gate schematic net list and the library cells to position the cells of the gate schematic net list in a two-dimensional format within a surface area of an integrated circuit die perimeter. The output of the place and route step is a two-dimensional physical design file which indicates the layout interconnection and two-dimensional IC physical arrangements of all gates/cells within the gate schematic net list.

According to the above prior art method, a separate internal data structure is used for each tool. This is because the tools are rarely if ever written by the same group; thus, the internal database representation for each tools is likely to differ from that of the other tools. Also, the most appropriate database implementation for the integrated circuit depends on the phase of the design process in which it is being used. For example, linked lists are commonly used to store cells in a netlist because that is the most obvious solution for logic synthesis purposes. In contrast, a KD tree is a more appropriate database format for the place and route tool.

This is time-consuming and processor-intensive (circuit specifications must be translated from one database format to another and another during the development process), disk-intensive (multiple databases each specifying the same circuit in different forms must be stored) and fragmented (tools cannot use the outputs of other tools, and a change to the circuit made by one tool is not reflected in the databases of the other tools).

SUMMARY OF THE INVENTION

The present invention has been made with the above problems of the prior art in mind, and a first object of the present invention is to provide a system for automated logic circuit design which is capable of storing and utilizing multiple levels of design data in a common database.

Another object of the present invention is to provide a system for automated logic circuit design which eliminates the need for translation of circuit descriptions between different design tools.

A further object of the present invention is to provide a system for automated logic circuit design which allows the output of tools in the design suite to be used by other tools.

Yet another object of the present invention is to provide a system for automated logic circuit design which allows design tools or the user to make area queries, i.e., a selection of a subset of objects based on their physical position, at various stages in the design process.

A still further object of the present invention is to provide a system for automated logic circuit design which permits the use of global simulation tools such as timing engines across all levels of design abstraction.

Another object of the present invention is to provide a system for automated logic circuit design which presents a unified model for timing, synthesis, placement and routing.

A further object of the present invention is to provide a system for automated logic circuit design which has high storage and run-time efficiency.

A still further object of the present invention is to provide a system for automated logic circuit design which has a consistent and easy to use programming interface.

A still further object of the present invention is to provide a system for automated logic circuit design which has an interface which is not dependent on other include files.

A further object of the present invention is to provide a system for automated logic circuit design which uses an object-oriented C++ programming style.

The above objects are achieved according to an aspect of the invention by providing an automated logic circuit design system which uses a common database to store design data at different states of the design process, including data-flow graphs, netlists and layout descriptions. In this way, the need to translate circuit descriptions between tools is eliminated, thus leading to increased speed, flexibility and integration. The common database includes entities, models, cells, pins, busses and nets. The data-flow graphs are stored as graphs, the nodes in a graph as cells, and the edges as busses. Physical design data is available by storing the cells in a model in a KD tree. This allows queries on cells in the netlist located in the layout within arbitrary areas.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects, features, and advantages of the present invention are better understood by reading the following detailed description of the preferred embodiment, taken in conjunction with the accompanying drawings, in which:

FIGS. 1–3 are dataflow diagrams of a circuit structure according to a preferred embodiment of the present invention;

FIG. 4 is a block diagram of the structure of a data model according to the present invention; and

FIGS. 5 and 6 are diagrams of the partitioning of a chip in correspondence with the data model.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EXEMPLARY EMBODIMENT

First, given a Verilog description of a circuit to be developed, the Verilog must be parsed to generate a data flow graph suitable for implementation in the data model. RTL parsers known in the art are preferably used for this purpose. The output from the RTL parser is a Verilog parse tree which is used to generate the data flow graph. Although well-known in the art, the structure of the parse tree is relatively complicated and, since detailed knowledge of it is not necessary for an understanding of the present invention, further description of the parse tree will be omitted for simplicity and brevity.

FIG. 1 shows an example of translation of the Verilog source code

```
always @(posedge clk)
begin
   out=in1+in2;
   if (c)
      out=in3;
end
```

into data flow elements. Here, in0, in1, in2, c and clk are input ports of an Entity (described below) and out is an output port of the Entity. An adder (an example of a Cell as described bellow) adds the values at Ports in0 and in1 and supplies the result to a merge block (another example of a Cell). If the value at Port c represents a logical true, the merge block supplies the value at Port in2 to a delay block (again, a Cell); if the value at Port c represents a logical false, the merge block supplies the output of the adder to the delay block. On the positive-going edge of the signal at Port clk, the delay block provides the value on its input to the output port out. The data flow graph having been generated, it may then be stored in the data model.

Before describing the data model in more detail, a few more examples are in order. FIG. 2 shows an example of the data graph resulting from the Verilog code

```
if (c1) begin
   out=in0;
end else begin
   if (c2) begin
      out=in1
   end else begin
      out=in2
   end
end
```

Here, if the value at input Port c1 is a logical true, the branch module connected to input Port in0 is enabled and provides its output to a merge module which is also enabled when c1 is true. If c1 is not true, the branch modules connected to Ports in1 and in2 are enabled to provide their outputs to other branch modules. One of the modules in this second tier is enabled when the value at input Port c2 is true and provides its output to another merge block. The other of the modules in the second tier is enabled when c2 is false and provides its output to the other merge block. Depending on the value of c2, one of those outputs is provided to the first merge block, and depending on the value of c1, one of the output is provided to the output port out via the delay element.

Finally, repetitive structures such as for, while and forever loops can be implemented in the data flow graph. FIG. 3 shows an example of the data flow graph implementation of the Verilog code

```
integer i;
for (i=0; i<5; i++)
begin
   a=a-4
end
```

Here, an entry node initializes a loop index with the value 0, adds 1 to the index and checks to see if the index is less than 5. If so, an exit node loops back for another pass. In tandem with this loop, an input variable a is received through another entry node and 4 is subtracted from it on each pass through the loop. When looping ceases because the index has reached 5, the current value of the input variable is presented at the exit node.

Certain optimizations can be performed on the data flow graph. For example, in the above loop structure the loop can be unrolled. That is, the graph portion representing the body of the loop can be replicated five times and the graph portion representing the loop index can be eliminated. For timing estimations and the like, virtual loop unrolling can be performed by estimating the number of iterations through the loop and using that number as a multiplier in delay calculations; the actual circuit replications can be done later.

Once the Verilog source is converted to a data flow graph, it can be implemented in the data model. Preferably, the data model is implemented using the C++ programming language or a similar object-oriented language. Since the construction, accessing and destruction of objects in such languages is well-known in the art, examples of specific commands for performing these operations will be omitted for brevity.

The topmost object in the data model 10 (shown in FIG. 4) is the Root object 20. The Root object 20 owns all other objects 30–70 and serves as a base to which everything else is attached. Also, the root 20 accommodates global attributes which are shared by all objects 20–70.

At the next level of the data model 10 is the Library object 30. Library objects 30 are used to organize entities 40. The preferred embodiment of the present invention includes at least two Libraries 30. The first stores information on the technology library to which the circuit under development will be mapped, and the second stores information on the circuit itself.

Next is the Entity object 40. An Entity 40 defines the interface of a design; that is, the set of ports 52 that the Entity 40 has. An Entity 40 may own Port objects 52. A Port 52 is used to represent the pin-out of an entity 40. Ports 52 cannot be individually created and destroyed, and can only be created when an Entity 40 is created. Each Port 52 has a direction (in, out, in/out) which is determined upon creation of the Entity 40. This rigidity promotes consistency between the Entity 40, its Models 50 and the Cells 62 bound to those Models 50.

As noted above, Entities 40 own Models 50. A Model 50 defines an implementation of an Entity 40; thus, multiple Models 50 within an Entity 40 represent different implementations of that Entity 40. Generally, these Models 50 are functionally identical to one another. For example, an Entity 40 in a technology Library 30 may have several Models 50 defining various eight bit adder cells having different power levels. Similarly, an Entity 40 in a target Library 30 may have several Models 50 which respectively define an abstract logic representation of a circuit, a gate-level implementation of it, a uniquefied representation, etc. The contents of each Model 50 is a net list of Nets 60, Cells 62 and Model Pins 64. All Models 50 and the Entity 40 have the same number of Pins 64 and Ports 52, and the Ports 52 have the same direction in the Entity 40 and over all Models 50; thus, it is relatively easy to replace one Model 50 with another from the same Entity 40.

Below the Models 50 are Cell objects 62. A Cell 62 represents a section of logic. Cells 62 may be primitive cells or non-primitive cells. Primitive Cells 62 have a predefined functionality associated with them. Preferably, the primitive Cells 62 include the following types:

CELL_AND—unlimited fan-in AND gate;

CELL_XOR—unlimited fan-in OR gate;

CELL_TRI—tri-state buffer

CELL_REG—sequential element

CELL_DC—don't care set

CELL_BREAK—break point cell; used to implement a "don't touch"; and

CELL_ONE—a constant one: an inverted or bubbled version is used for a constant zero.

In contrast to primitive Cells, the functionality of non-primitive Cells is defined by technology Models 50 to which they are bound. That is, a Cell 62 may describe a portion of the circuit under development and belong to a Model 50 in a target Library 30. However, it will be associated with (preferably by pointing through a cell type attribute or the like) a Model 50 in a technology library 30 which defines its functionality and general characteristics.

Non-primitive Cells 62 may be created as bound Cells; alternatively, they may be created as unbound Cells and later

bound to a Model 50. This may be done by specifying the Cell's name; by specifying pin-to-pin correspondence vectors; and by binding the Cell 62 to an undefined Model 50 and later matching the Model 50 to an actual one. Additionally, a bound Cell 62 can be rebound to a different Model 50 within the same Entity 40.

Each Cell 62 includes a number of parameters called members which specify certain features of the Cell 62. These include the cell's name, a pointer to the technology Model 50 to which it is bound, a list of Pins 64 which it owns, its parent Entity 40, and coordinates of the Cell 62 within the chip layout.

Net objects 60 make connections between pins. The pins may be Model pins 64 or Cell pins 70. A Net 60 does not own Pins 64 and 70, and deleting the Net 60 will leave the pins 64 and 70 disconnected. Pins 64 and 70 may be grouped into Busses 80 (in fact, every variable in the Verilog code will be represented as a Bus). Since Pins 64 and 70 are the most common object in almost any circuit representation, it is important to reduce the amount of storage for each Pin 64 and 70 as much as possible while maintaining easy accessibility. For this reason, Pins 64 and 70 are preferably stored in small arrays and associated with indices.

Nets 60 also have members, such as the Net's name, a list of Pins 64 and 70 which it connects, and a list of rectangles through which it passes in the placement layout. Pin members include the Pin's name, the Model 50 or Cell 62 to which it belongs, and the Net 60 to which it is connected.

Each object 20–70 may have a number of attributes. Each attribute has a name and a value of a type int, short, float, double, char* and void*. One example of an object attribute is an inversion attribute or "bubble" which specifies whether a Cell input or output (or Net 60) is asserted high or low. Other examples of object attributes are object name, firing information, references to the Verilog code defining the object, etc.

Iterators are procedures used to access objects within the data model. As is known in the art, an iterator traverses the model and each time it is called, returns a pointer to the next object of a particular type. For example, a Model iterator would, when successively called, return pointers to each Model 50 within the data model. The preferred embodiment of the present invention provides "safe" and "unsafe" iterators, where unsafe iterators return all objects of the specified type, even if they have been added during the iteration process, and safe iterators omit objects added during the iteration. In this way, although the safe iterators are slightly slower than their unsafe counterparts, they can avoid program crashes, errors and exceptions, and other undesirable outcomes.

Before synthesis and timing can take place it is often necessary to uniquefy the data model. This involves binding each Cell 62 to its own individual technology Model 50. This simplifies the synthesis process in that changes made to one technology Model 50 will affect only the Cell 62 which is bound to it, and no others. Also, after uniquefication it is possible to traverse the data model both up and down, since each object has a unique parent and child. Typically, uniquefication is done by making a copy of a technology Model 50 for each Cell 62 which is bound to it and associating one of the cells 62 to each copy.

After the data model has been uniquefied, it may be ungrouped, i.e., macro-level cells can be replaced with their primitive components. Alternatively, processes may handle the data model with virtual ungrouping by "looking through" the macro-level cells to process their primitive cell constituents.

With this understanding of the structure of the data model in mind, implementation of a Verilog-derived data flow graph in the data model will now be described. For each module in the Verilog description there will be one Entity **40** and one Model **50** (hereinafter collectively referred to as a graph). The ports for the Entity **40** correspond to the ports in the Verilog module. Ports **52** in the graph have a bit width, and there will be a separate Pin **64** and Net **60** (the group of Nets **60** for the Port **52** forming a Bus) in the graph for each Verilog port.

For each node in the Verilog module, a Cell **62** will be made in the graph. Initially the Cells **62** will be unbound. As described above, given the Cell type and the Pins **70** of the Cell **62**, a Model **50** for the Cell **62** to be bound can be generated later.

Each Model **50** is preferably implemented as a KD tree as follows. First, the circuit under development is divided into a number of sections each corresponding to a rectangular section **100** of the available chip area as shown in FIG. **5**. The partitioning of the circuit can be directed by the user; however, it is preferably automatically done by the system so that the circuit is evenly distributed over the entire chip area. Each node or leaf **210** of the KD tree **200** shown in FIG. **6** corresponds to a cutline **110** of the rectangles **100** and may have appended thereto a linked list **220** of all cells **62** which lie on that cutline **110**. Non-leaf nodes **210** in the KD tree **200** each have two child nodes **210**, with the left child **210** corresponding to the region of the chip on one side of the cutline **110** and the right child **210** corresponding to the region of the chip on the other side of the cutline **110**. Similarly, the child nodes **210** may have linked lists **220** of cells on their cutlines **110** and child nodes **210** of their own.

It should be noted that the leaf nodes **210** will contain most of the circuit information, since the non-leaf nodes **210** will only have information on those cells touching their corresponding cutline.

As noted above, the initial distribution of Cells **62** over the chip area is preferably done automatically by the system and in that case may be done through the use of various algorithms which will readily suggest themselves to those skilled in the art. The result of this process is a model with mostly logical information on its constituent elements but with a coarse framework of physical placement and routing information, e.g., cell areas, initial placements, etc. In later steps of the development process described below, the physical information will be refined and augmented within the original data model. In this way, the addition of rough physical layout information to the initial logical description enables the smooth transition of the circuit through the development process, thereby enabling sharing of tool outputs, use of common diagnostics and the like.

Further, once RTL synthesis is complete and the data model is flattened, it may be copied and used as a baseline for formal verification and the like. Since a common model structure is used, there is no need to translate the pre-logic synthesis version of the circuit into a format suitable for use by the verification tool.

As the development process progresses, the KD tree **200** may become unbalanced due to an excessive number of additions or deletions in one area, or due to poor initial distribution. This can be compensated for by manual rebalancing by the user or by a user-initiated procedure, but preferably is done automatically by the system.

Once the data model has been constructed in this way, it may be used for both logic synthesis, i.e., gate-level implementation, etc., and physical synthesis, i.e., placing and routing. This is because the data model includes all of

the information necessary for logical synthesis operations, i.e., cell functionality, net connections, etc., as well as all information necessary for physical synthesis operations, i.e., areas, physical positions, etc.

Another advantage of the data model arises from its correspondence with the actual physical chip layout. Since each node of the KD tree **200** corresponds to a cutline **110** and has associated with it the cells on the cutline and information on where its child nodes are within the chip area, portions of the circuit in specific physical areas can be queried, tested and manipulated without the need to read the entire data model into active memory from disk storage, as is the case with prior art net lists. For example, assuming a user wanted to work with only the lower right hand corner of the chip, the system could traverse the KD tree to reach the topmost node corresponding to that area. Then, that node, its children, netlists and the like would be read into active memory from disk and manipulated. The user may even be able to manually direct placement of cutlines **110** at certain points to frame a particular area of interest. The system may then adjust the KD tree accordingly to accommodate the new arrangement. This area query technique is possible whether the circuit is in its final placement and routing stages or fresh from Verilog synthesis.

Although only a portion of the entire data model need be read into memory, the complete set of Nets **60** is typically maintained in memory. This is because the Nets **60** are necessary for purposes such as delay estimation and the like that are performed frequently, and it is easier to retain all Nets **60** in memory rather than repeatedly read them into memory. Thus, once a specific area has been designated for querying, the Nets **60** corresponding to that area must be identified. This is done by identifying the Nets **60** connected to each of the Pins **64**, **70** within the selected area. The remaining Nets **60** can be eliminated from consideration during the area query. Nets **60** which have some, but not all, Pins **64**, **70** within the query area can have the missing pins represented by a stub pin. Finally, Nets **60** which have all of their pins within the query area can be handled as are other objects within the selected area.

Further, during the area query process, Nets **60** which are entirely contained within the selected area can be optimized out or otherwise modified; however those nets having portions outside the query area, i.e., those with stub pins, cannot, since the effect of modification of elimination of these Nets **60** on the remaining circuit portions is unpredictable.

Further, since the logical and physical aspects of the circuit are integrated into a single data model from the start, deviations from the classic logical synthesis/physical synthesis partition can be made. For example, the inclusion of buffers for load handling and timing purposes is normally done as part of the logical synthesis process; however, using a common data model for all aspects of the development process allows the placement of buffers to be delayed until later during the placement process, when layout information is more definite and precise. The above description of the preferred embodiment of the present invention has been given for purposes of illustration only, and variations thereof will be readily apparent to those skilled in the art. For example, although Verilog has been used as the preferred language for initial input of the circuit under development, other appropriate hardware description languages may of course be used. Also, although implementation of the data model using object-oriented C++ techniques has been disclosed, other programming languages and paradigms may also be workable. Similarly, alternative object hierarchies may be used. Such variations fall within the scope of the present invention. Thus, the scope of the present invention should be limited only by the appended claims.

What is claimed is:

1. A common data model representing a circuit that will be fabricated on an integrated circuit chip comprising:

a data representation including a plurality of objects that together represent the circuit, certain ones of the objects including a netlist portion that represents a corresponding portion of the circuit, and each of the objects:

being logically correlated to at least one other object so that all of the objects describe the circuit; and

each of the objects, once associated with a physical location is adapted for subsequent retrieval using an area query corresponding to the physical location.

2. The model according to claim 1 wherein the physical location association of objects is implemented using hierarchical partitioning.

3. The model according to claim 2 wherein the bierarchical partitioning is implemented using a tree.

4. The model according to claim 3 wherein the circuit is represented within an area, with a plurality of cutlines that partition the area into a plurality of rectagles.

5. The model according to claim 4 wherein the tree contains a plurality of leaf nodes, and each of the leaf nodes corresponds to one of the cutlines.

6. The model according to claim 5 wherein the tree includes a linked list that identifies each cell that lies on a particular one of the cutlines.

7. The model according to claim 5 wherein the tree contains a plurality of non-leaf nodes, each of the non-leaf nodes associated with one of the leaf nodes, and each of the non-leaf nodes, containing at least two child nodes, each child node corresponding to an area on an opposite side of the cutline associated with the one leaf node.

8. The model according to claim 3 wherein certain of the objects represent cells.

9. The model according to claim 3 wherein certain of the objects represent a net or a part of a net.

10. The model according to claim 3 wherein certain of the objects represent pins.

11. The model according to claim 1 wherein the each of the objects corresponding to each of the physical locations is maintained in an active memory.

12. The model according to claim 11 wherein the subsequent retrieval of objects corresponding to the physical location of the area query causes the retrieval of all objects associated with the physical location to be retrieved into an active memory.

13. The model according to claim 12 wherein the retrieval of all objects associated with the physical location is from a disk storage.

14. The model according to claim 1 wherein the model allows for insertion of cutlines to frame a particular area of interest.

15. The model according to claim 1 wherein the area query takes place either immediately after synthesis or during final placement and routing.

16. The model according to claim 1 wherein the model is configured to allow for a logical query to take place.

17. The model according to claim 16 wherein the logical query of one object provides at least another object that is logically related to the one object.

* * * * *

EXHIBIT 5

FULLY REDACTED

EXHIBIT 6

US006505328B1

(12) **United States Patent**
Van Ginneken et al.

(10) **Patent No.: US 6,505,328 B1**
(45) **Date of Patent: Jan. 7, 2003**

(54) **METHOD FOR STORING MULTIPLE LEVELS OF DESIGN DATA IN A COMMON DATABASE**

(75) Inventors: **Lukas P. P. P. Van Ginneken**, San Jose, CA (US); **Patrick R. Groeneveld**, San Jose, CA (US); **Wilhelmus J. M. Philipsen**, Phoenix, AZ (US)

(73) Assignee: **Magma Design Automation, Inc.**, Cupertino, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/300,540**

(22) Filed: **Apr. 27, 1999**

(51) **Int. Cl.**^{7} **G06F 17/50**
(52) **U.S. Cl.** **716/7**; 716/8; 716/12
(58) **Field of Search** 716/1–21

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,187,668	A	*	2/1993	Okude et al.	716/8
5,212,650	A	*	5/1993	Hooper et al.	716/18
5,313,615	A	*	5/1994	Newman et al.	716/11
5,432,707	A	*	7/1995	Leung	716/2
5,452,226	A	*	9/1995	Hooper et al.	716/18
5,487,018	A	*	1/1996	Loos et al.	716/11
5,519,627	A	*	5/1996	Mahmood et al.	716/18
5,541,849	A	*	7/1996	Rostoker et al.	716/18
5,623,417	A	*	4/1997	Iwasaki et al.	716/18
5,666,288	A	*	9/1997	Jones et al.	716/17
5,696,693	A	*	12/1997	Aubel et al.	716/8
5,699,265	A	*	12/1997	Scepanovic et al.	716/10
5,726,902	A	*	3/1998	Mahmood et al.	716/6
5,727,187	A	*	3/1998	Lemche et al.	716/18
5,757,657	A		5/1998	Hathaway et al.	710/9
5,761,664	A		6/1998	Sayah et al.	707/100
5,764,534	A		6/1998	Goetting	716/11
5,818,729	A	*	10/1998	Wang et al.	716/9
5,841,663	A	*	11/1998	Sharma et al.	716/18
5,864,487	A	*	1/1999	Merryman et al.	716/6
5,956,497	A	*	9/1999	Ratzel et al.	716/1

(List continued on next page.)

OTHER PUBLICATIONS

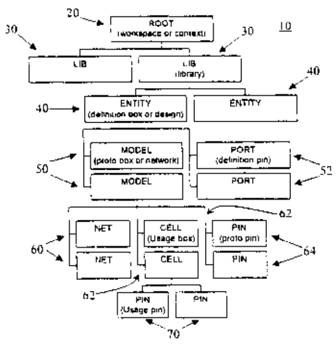
Dutt ("Generic component library characterization for high level synthesis", Proceedings of the Fourth CSI/IEEE International Symposium on VLSI Design, 1991, Jan. 4, 1991, pp. 5–10).*
Dion, J. and Monier, L.M.; "Countour: A Title–based Gridless Router," *WRL Research Report 3/95*, Digital Western Research Laboratory.
Dion, J. and Monier, L.M.; "Recursive Layout Generation," *WRL Research Report 2/95*, Digital Western Research Laboratory.
Hwang., J., et al., "Generating layouts for self–emplementing modules"; Intl Workshop on Field Programmable Logic and Applications, FPGAS, GB, Abingdon, Aug. 31, 1998, pp. 525–529.
Singhal, A., et al., "Object oriented data modeling for VLSI/CAD," Proc. of the 8th Intl Conf. on VLSI Design, New Delhi, India, Jan. 4–7, 1995, pp. 25–29.
Fcanha, H.S.;, "Data astructtures for physical representation of VLSI," Software Eng'g Journal, GB, IEE. London, vol. 5, No. 6, Nov. 1, 1990.

Primary Examiner—Matthew Smith
Assistant Examiner—Phallaka Kik
(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop LLP

(57) **ABSTRACT**

An automated logic circuit design system uses a common database to store design data at different states of the design process, including data-flow graphs, netlists and layout descriptions. In this way, the need to translate circuit descriptions between tools is eliminated, thus leading to increased speed, flexibility and integration. The common database includes entities, models, cells, pins, busses and nets. The data-flow graphs are stored as graphs, the nodes in a graph as cells, and the edges as busses. Physical design data is available by storing the cells in a model in a KD tree. This allows queries on cells in the netlist located in the layout within arbitrary areas.

17 Claims, 4 Drawing Sheets



SYN0012146

May-31-02 04:41pm From-PILLSBURY WINTHROP LLP SV V#2 +6502334545 T-603 P 001/005 F-750

PILLSBURY WINTHROP LLP

2550 HANOVER STREET PALO ALTO, CA 94304-1115 650.233.4500 F: 650.233.4545

FACSIMILE

Total Pages (including cover): 5

LONDON
LOS ANGELES
NEW YORK
NORTHERN VIRGINIA
ORANGE COUNTY
SACRAMENTO
SAN DIEGO
SAN FRANCISCO
SILICON VALLEY
SINGAPORE
STAMFORD
SYDNEY
TOKYO
WASHINGTON DC

Date:	May 31, 2002	Must Be Sent By:	
From:	David A. Jakopin	Phone No:	(650) 233-4790
User No:	13016	C/M No:	054355-0253032
To:	*EXAMINER P. KIK* *ART UNIT 2825*		
Company:	*U.S. Patent and Trademark Office*	Fax No:	703-872-9319

Comments:

REQUEST FOR CONTINUED EXAMINATION (RCE)
AND
PRELIMINARY AMENDMENT

Van Ginneken
Serial No. 09/300,540

FAX COPY RECEIVED
MAY 31 2002
TECHNOLOGY CENTER 2800

If you have not properly received this fax, please call (650) 233-4500. Thank you.

Operator: ______________ Time Sent: ______________ Batch ID: ______________

Received from < +6502334545 > at 5/31/02 7:42:58 PM [Eastern Daylight Time]

Amdt C

DETAILED ACTION

Response to Preliminary Amendment

1. This Office Action responds to Applicant filing of RCE and preliminary amendment filed on 5/31/2002. Claims 48-64 are pending, wherein claims 1-47 have been cancelled and claims 48-65 are newly added. Claims 48-65 have been examined and are allowed, wherein claims 48,58-63 are subject to the following Examiner's Amendment.

Continued Examination Under 37 CFR 1.114

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 5/31/2002 has been entered.

Examiner's Amendment

3. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with David A. Jakopin (Reg. No. 32,995) on 7/12/2002.

The application has been amended as follows:

<u>**In the claims:**</u>

As per **claim 48**, "can subsequently be retrieved" (lines 8-9) has been replaced with –is adapted for subsequent retrieval--.

As per **claim 58**, "that can be operated upon" (line 2) has been deleted.

As per **claim 59**, "to be retrieved" (line 3) has been deleted;
"that can be operated upon" (lines 3-4) has been deleted.

As per **claim 60**, "object" (line 1) has been replaced with –objects--.

As per **claim 61**, "the insertion" (line 1) has been replaced with –insertion--.

As per **claim 62**, "can take" (line 1) has been replaced with –takes--.

As per **claim 63**, "configures" (line 1) has been replaced with –configured--.

Allowable Subject Matter

4. **Claims 48-64** are allowed.

5. The following is an examiner's statement of reasons for allowance:

As per **claims 48-64**, the independent claim 48, which the claims depend, recites the common data modeling representing a circuit that will be fabricated on an integrated circuit, comprising the inventive feature of having each of the objects, once associated with a physical location, is adapted for subsequent retrieval using an area query corresponding to the physical location, as part of the data model as claimed, which corresponds to Applicant's specification, page 6, line 14 to page 16, line 4. The prior arts made of record teach various methods of modeling circuit design, including using hierarchical tree structures and query methods (see especially **Scepanovic et al.**, US Patent No. 6,154,874, especially abstract and col. 7, line 53 to col. 8, line 67; **Wang et**

al., US Patent No. 5,818,729, especially abstract and Figs. 10A, 10B; **Cleereman et al.**, US Patent No. 5,960,184, especially Fig. 9; col. 7, line 56 to col. 8, line 5; **Mohan et al.**, US Patent No. 6,216,258, especially col. 2, lines 10-37; **Mahmood et al.**, US Patent No. 5,726,902, especially abstract; Fig. 6; col. 12, line 56 to col. 13, line 65; **Hooper et al.**, US Patent No. 5,212,650, especially abstract; Figs. 1-4). However, none of the prior arts made of record teach the inventive features of using the area query as part of the common data model as claimed. Accordingly, the claimed invention is novel and un-obvious over the prior arts made of record.

Conclusion

6. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Phallaka Kik whose telephone number is 703-306-3039. The examiner can normally be reached on Flexitime.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Matthew S. Smith can be reached on 703-308-1323. The fax phone numbers for the organization where this application or proceeding is assigned are 703-872-9318 for regular communications and 703-872-9319 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-308-1782.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

703-872-9318 (for Before-Final) and 703-872-9319 (for After-Final) for formal communications intended for entry,

Or:

(703) 746-4111 (for informal or draft communications, please label "PROPOSED" or "DRAFT" and let the examiner know prior to faxing)

Hand-delivered responses should be brought to Crystal Plaza 4, 2201 South Clark Place, Arlington, VA 22202, Fourth Floor (Receptionist).

PK [illegible]
July 12, 2002

[illegible]

[illegible] SMITH
[illegible] EXAMINER
[illegible] CENTER 2800

EXHIBIT 7

The PowerPC*601* Design Methodology

Tim Brodnax, Mike Schiffli and Floyd Watson
IBM Corporation
11400 Burnet Road, 9730
Austin, Texas 78758

Abstract - *To produce a marketable PowerPC* microprocessor on a short development schedule, the appropriate balance was needed between a fully customized methodology and a standard cell approach. We compiled a single model of each logic partition separately for logic synthesis and for simulation. Synthesis transferred the model to a gate-level implementation and applied timing correction transforms to reach the users' timing assertions. Floorplanning and wiring was assisted by an automated tool. Billions of cycles were run in testing the implementation against the PowerPC architecture specification. Behaviorals were written and built into the model to test the chip as it might appear in a system configuration. Careful adherence to this methodology led to a successful first pass of silicon, leaving the second iteration for additional customer requests.*

I Introduction

When the alliance was signed between Motorola, IBM and Apple Computer, Inc. to develop state of the art microprocessors, the strategy created some unique pressures. We needed an initial offering on the market on a very short development schedule. Moreover, the initial processor needed to be very attractive according to cost / performance trade-offs. The PowerPC 601 exceeds our goals by each of these criteria. This paper describes the design methodology which was essential to this development with specific details about logic design, physical design and the simulation approach.

i Common Database

The PowerPC 601's design tools came from evolutionary changes to the tool set used to build the previous IBM RISC System/6000* chips [2],[3],[4]. Our proprietary tool set now runs entirely on our workstation platform, utilizing a common database where reasonable.

The 601's logic designers described the chip in a proprietary high-level language called Design Structure Language (DSL). The DSL compilers accept hardware constructs in a program-like manner and support many levels of hierarchy for a macro design approach. Fig. 1 shows the flow of operations for the two distinct tasks - behavioral verification and logic design to physical design.

ii Standard Cell Simplicity with Full-Custom Densities

With many challenging functions in our dataflow, we used customized, hand-placed memory elements, multiplexors and arithmetic functions. These customized, manually-placed circuits (designated Off-The-Shelf or OTS) led to optimal timing and density. The control logic was developed by allowing synthesis (we use IBM's BooleDozer* [1]) to map the high-level, functional model for each Random Logic Macro (RLM) to primitive gates

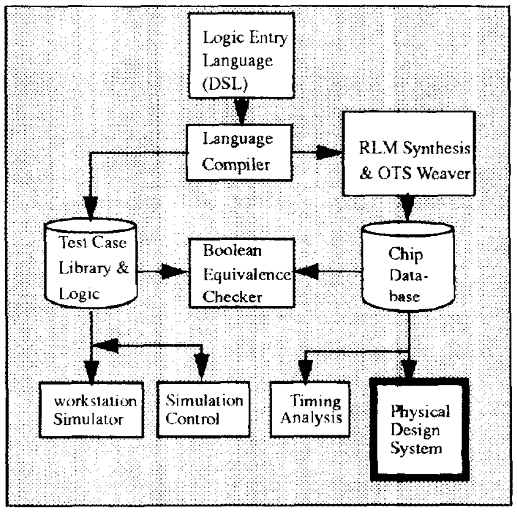


Fig. 1 601 Design Methodology

1063-6404/93 $03.00 © 1993 IEEE

in the technology library. Logic designers improved the timing of critical paths by reworking the function, hiding a desired implementation from synthesis or altering synthesis timing assertions.

iii Client-Server Computing Environment

To take full advantage of the workstation platform, the users need increased access to a large workstation network. Load-balancing, job routing software was developed to distribute synthesis and simulation tasks. This has proven to be an efficient way to maximize our usage of the available computers without requiring each designer to labor through many networking details. The designer dispatches his application to the server, which is routed to an available client.

II Logic Design

i Logic Synthesis

As described above, 601 control logic was partitioned into 38 RLMs. BooleDozer implements the logic in a technology independent form, then maps it into the user-specified technology library. The resulting logic will be optimized for area.

A number of transforms are applied to the logic to attempt to realize the timing relationships in the users' assertion files. These transforms perform simple tasks - increasing power (and size) of a gate, duplicating logic to alleviate fanout problems, swapping pins, or remapping a function to the faster gates (ex: AND/OR to NAND/NAND). The assertion files are critical. If these assertions are not aggressive enough, unacceptable paths may still exist when RLMs are tied together. Overly aggressive assertions may contradict each other, leaving timing problems and causing extra logic which puts additional strain on the floorplan. Late in the PowerPC 601 development, the Global Assertion File Generator partitioned chip timing paths into RLM assertions which saved roughly 10% of our RLM cells (a very needed improvement for our floorplan team) and helped reduce some of the critical paths. Synthesis also provided two ways for the user to 'hide' an intended implementation from synthesis. This feature was used sparingly.

ii Assembling the elements

BooleDozer provide a gate level implementation for each RLM. Simultaneously, Weaver takes all chip DSL and maps the OTS (which requires no synthesis) into one file with the OTS connected and prototypes for the RLMs. These prototypes are footprints which contain the signal I/O for each RLM. A 'flattener' then ties the OTS gates with the individual RLMs into one 'flattened' description of the entire chip.

iii Timing Analysis

Once the design has been flattened, it is analyzed with our static timing tool, Early Timing Estimator (ETE). In addition to the chip's gate level description, ETE accepts capacitances and Resistance-Capacitance delays (RCs), chip I/O timing assertions and the clock waveform. Estimates based on fanouts were used for the capacitances and RCs before the real data was available from the floorplan. ETE uses these design-specific inputs along with macro timing equations for the technology library to calculate cumulative delays between latch/memory elements. With a description of the clocks and these delays, ETE can identify paths which are failing to meet the required timing and the amount of excess delay. The logic designers were then faced with the traditional methods of fixing timing problems - altering the logic, changing fanouts, or altering the floorplan to improve critical capacitances and RCs.

III Physical Design

The PowerPC 601 was fabricated using IBM's proprietary .6um minimum feature process technology. The technology includes 4 levels of signal interconnect, a silicide layer for local interconnect, and a fifth coarse level of metal for connecting chip I/O's to the module substrate. As a cost savings measure, one wiring plane was deleted when wiring the chip. The chip die, seen in Fig. 2, contains 2.8 million transistors in 120 square millimeters. The chip is packaged in a 304 pin ceramic Quad Flat Pack using IBM's C4 flip chip technology.

Physical design for the PowerPC 601 microprocessor was done with IBM's VIDAS tool set. VIDAS is a top-down design system for assembling semi-custom VLSI chips. It utilizes as input a logical description of the chip and a library of circuit elements. The logical description is in a netlist format and the circuit library is in a human readable form that describes the key physical features of the circuit elements. The tools are organized such that global floorplanning can proceed in parallel with global logic design and circuit library design.

i Primitive Macro Design

The circuit library used for the PowerPC 601 chip was designed jointly in Austin, Texas and at IBM's Burlington, Vermont site. The circuit library was divided into two portions - RLM circuits and OTS components.

The RLM circuits (low-level functional blocks such as inverters, NAND gates, and AndOrInvert circuits) were used to construct the control logic for the chip dataflow. The OTS components include more complex functions such as data registers, multiplexors, adders and memory arrays. These components are highly customized. The 32KB Cache is the best example of the densities possible in this arrangement; it contains almost two thirds of the transistors in roughly one third of the chip area (Fig. 2) The OTS components were designed in 4-bit increments and stitched together to obtain a desired component width. The density of the circuit designs benefitted from the use of the silicide local interconnects between diffusions and polysilicon and studded contacts between wiring levels. Another aspect of the macro design is the timing characteristic of the function. The macro designer runs a transistor-level simulator to generate the ETE library discussed in the previous section.

ii Floorplanning

The global floorplanning began after delivery of an initial description of the circuit library and a preliminary netlist description of the chip. The initial description of the circuit library contained necessary information about the circuit macros such as pin locations, power connections, and wiring blockages. The detailed work on each of the circuit macros is free to proceed as long it does not violate the original specifications. CPLACE, a stack partitioning and placement optimization tool, uses the circuit library description and the logical netlist to make an initial global placement. Several dozen passes through CPLACE are made to achieve a satisfactory initial placement. An interactive placement tool, VIGR, is used to view the interconnectivity of the CPLACE floorplan and to make manual placement adjustments. The global power distribution is formed by connecting macro power supplies together with a 3 level mesh.

The initial wireability analysis of the chip is performed via WEVAL, a wiring evaluation tool. WEVAL performs a coarse maze routing of all of the global chip nets. WEVAL arrives at a solution much faster than a detailed router because it does not output a specific wiring solution. Instead, it divides the chip into tiles of a user specified size and calculates the approximate congestion through each tile. If the initial floorplan exhibits wireability problems, the design flow is started over at CPLACE. When a suitable wiring solution is indicated by WEVAL, the design flow continues. The wiring solution from WEVAL is used to assign blockage data and pin assignments to the random logic control macros. At this point, the global design is divided into random logic macro design, global design, and top-metal design. Each RLM is pushed through the design flow as though it were a chip, using the constraints forced onto it by the global design. The hierarchical nature of the design process was also exploited by separating the design of the floating point unit from the global design flow.

The global routing is performed by LGWIRE, a global routing program. LGWIRE reads description of the circuit library elements and outputs a completed wiring solution. The top-metal design is manually entered using the coarse uppermost layer of interconnect provided by the process technology. The top-metal design is hand-drawn to connect chip I/O signals to module pins and to complete the global power grid. Any areas not needed for connecting chip I/O's and global power buses is returned to the chip for wiring global nets during subsequent LGWIRE passes.

After the chip was successfully wired, global wiring capacitances and resistances were extracted and provided to the logic design team to fold into final timing calculations. At intermediate points in the design, RCs were estimated based upon the current placement and a small multiplying factor to allow for non-manhattan wiring. This allows timing decisions to be made during the logic design phase. Further tuning is accomplished by manual rerouting critical nets to reduce parasitic RCs.

Verification of the completed physical design is accomplished with a set of hierarchical verification tools. Individual library circuit elements and random logic macros are checked to verify groundrule and logical-to-physical correctness. Shadow shapes are generated to group similar features in the circuit macros and to lessen data volume. The interaction of the global wiring and I/O interconnects are then checked against these shadow shapes.

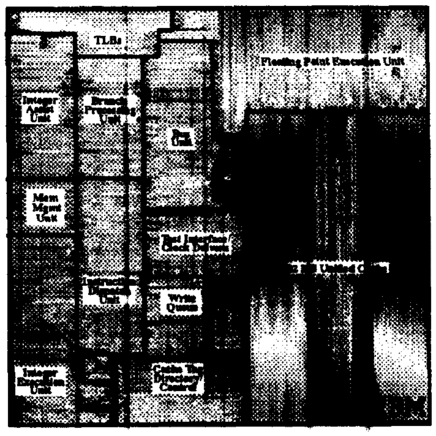


Fig. 2 601 Die Photograph

IV Simulation

i Unit-Level Verification

Each unit of the design was verified using our cycle simulator and some custom code to trigger the unit-under-test's inputs and then check the outputs against known values. Initially, subunits within each unit were verified until each unit shown in Fig. 3 was verified. After several months of unit and multiunit verification, all units were combined into a system model.

ii System-Level Verification

The system model consists of the complete PowerPC 601 design as well as behaviorals which were written in order to test the chip as it might appear in a system configuration. The behaviorals were coded in such a way as to allow any timing patterns to be generated, either randomly or precisely on the input pins. To solve the problem of generating the volume and variety of tests needed to verify the PowerPC 601 design, two approaches were taken. First, Architecture Verification Programs (AVPs) were used to test the instruction set implementation against the PowerPC architecture specification. A software reference model was used to generate millions of tests, both randomly and manually. Second, Implementation Verification Programs (IVPs) were used to test the hardware implementation and special hardware features. Specific tests were written for each unit shown in Fig. 3. These were then regressed for every model change. The various timing patterns as described above were applied to the PowerPC 601 input pins while these AVPs and IVPs were simulating billions of cycles, taking advantage of the clustered workstations. The multiprocessor environment was also extensively tested with several PowerPC 601s built into a multiprocessor simulation model.

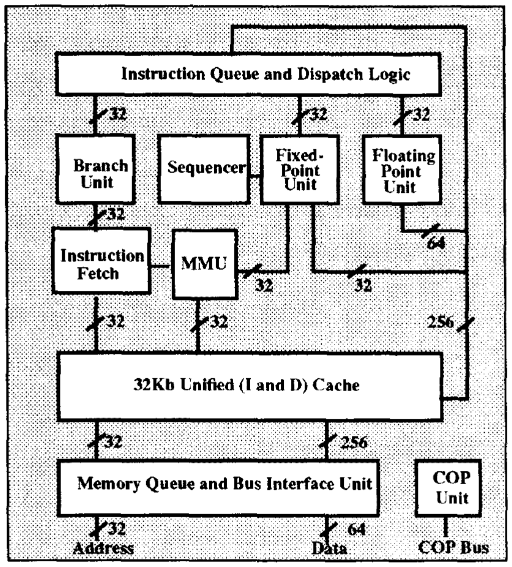


Fig. 3 601 Dataflow

iii Gate-Level simulation

Behavioral simulation is clearly the most efficient method to test an evolving design's pipeline interaction. Unfortunately, there exists a limited set of tests which require some timing data from the actual implementation. We ran AUSSIM (AUStin SIMulator) with handwritten tests to test a number of operations: checking for glitches, ensuring the test features not enabled during normal operation, Power-on-reset, and asynchronous I/O.

V Conclusion

We continue to develop this design system for other projects. As stated above, it allowed us to deliver our first implementation of the PowerPC in under a year. The core of this design system is now commercially available and should be considered by all integrated circuit design centers[5].

We achieved a successful first pass of silicon, leaving the second iteration for additional customer requests. At 66 MHz, the PowerPC 601 attains 60 SPECint92* and 80 SPECfp92*, based on estimates from simulations. The success of the design team in utilizing state of the art CMOS technology in a compact, single chip gives the 601 an outstanding price/performance ratio and makes it well suited for a wide range of system designs.[6][7]. We have exceeded the performance of larger processors which use the more complicated (and expensive) BiCMOS technology. We feel this is due to creative, diligent work within the structure of our design methodology. This combination allows us to make this chip available at a fraction of the price, size and power of processors with comparable performance.

VI Acknowledgments

This was a unique project with aggressive function, performance and schedule goals. It required the cooperation of many people from Apple, Motorola and several divisions within IBM. In addition, the project was viewed as a barometer for the health of the overall relationship of the alliance. In the end, the team stepped

up to these expectations and met all of the project goals.

The authors would like to acknowledge all of the logical design, physical design, methodology support and verification engineers that worked on the 601 processor. This project would not have been possible without the hard work and personal sacrifices made by these people. In particular, John Muhich and Chuck Moore provided much of the architectural and organizational framework for the chip. Ray East, Doug Balser, Mark Papermaster, Susan Tiner, Michael Becker and David Tuttle provided exceptional management focus and vision. Chris Olson provided key expertise in the implementation of the Floating Point Unit. Brian Vicknair, Tim Elliott and Ed O'Neil designed the Cache Unit. Jack Wilson and Mike Vaden were the designers of the Fixed Point Unit and the Memory Management Unit. Terence Potter and Tan Chu designed the Branch Processor and Instruction Fetcher. John Zimmerman and Tony Penaloza were responsible for the Sequencer Unit and the microcode. Rob Reese, Michael Garcia, Michael Allen and Larry Merkel did an outstanding job on the Bus Interface Unit. Richard Billings, Eddie Rodriguez and Casey Nguyen provided expertise in the design of the COP and in the test features of the chip. Charlie Roth worked with Tim Brodnax to drive the logic tuning and timing analysis effort. Mike Schiffli was fortunate to have the services of Joe Werner, Frank Palermo, Scott Glenn, Chris Byant, Brian Reynolds, Larry Powell, Tom Sartorius, John Meyer, Peter Voldstadt, Bob Young, Hai Vu, John Bunda, and Robert Keist applied to the design verification effort. Gavin Meil and Jeff Brooks took care of all aspects of the hardware verification. P.T. Patel performed the early sizing estimates for the chip and provided invaluable assistance in leading the chip integration team - Floyd Watson, Joaquin Fentanes and Brian Zoric. Phuong Bhan, Bart Martinec and Steve Quay placed primitive cells within the RLMs. Steve Posluszny adapted the synthesis routines written by IBM Research to our needs. This list is still incomplete; contributions to this design system have been made over several years by a number of talented programmers, engineers and managers at several IBM sites.

VII References

[1] D. Kung, R. Damiano, and T. Nix, "BDDMAP: A Technology Mapper on a New Converging Algorithm," in Poceedings of the 29th Design Automation Conference, June 1992.

[2] C. H. Feistel, H. Hoffman, G. B. Long, and G. A. Nusbaum, "Structured System Design and Verification," IBM RISC System/6000 Technology, SA23-2619, IBM Corporation, 1990, p.86.

[3] J. W. Cagle, P. T. Patel, B. I. Waters, and P. G. Villarrubia, "Semi-Custom Chip Design Methodology," IBM RISC System/6000 Technology, SA23-2619, IBM Corporation, 1990, p.92.

[4] E. Seewann, S. L. Runyon, R. K. Montoye, Q. Nguyen, and J. C. Ridings, "VLSI Circuit Design for the RISC System/6000 Processor" IBM RISC System/6000 Technology, SA23-2619, IBM Corporation, 1990, p.98.

[5] IBM Presentation at the 30th Design Automation Conference, June 1993.

[6] Moore, C. R., "The PowerPC 601 Microprocessor," Proceedings of COMPCON 1993, February 1993.

[7] Case, Brian, "IBM Delivers First PowerPC Microprocessor", Microprocessor Report, October 28, 1992 p.1..

* PowerPC, PowerPC 601 and BooleDozer are trademarks of International Business Machines Corporation.

* IBM and RISC System / 6000 are registered trademarks of International Business Machines Corporation.

SPECint92 and SPECfp92 are trademarks of Standard Performance Evaluation Corporation.

The authors may be reached at brodnax,schiffli or floydw%ibmoto.com@aimgate.sps.mot.com

TAB 8

FULLY REDACTED

EXHIBIT 9

Design Flow and Methodology for 50M gate ASIC

Alok Mehrotra, Lukas van Ginneken, Yatin Trivedi
Magma Design Automation Inc.
Cupertino, CA 95014

Abstract: This paper presents a methodology for full chip RTL timing closure for very large ASIC's. The methodology is based on the concept of a "Silicon Virtual Prototype". The methodology is based on the scalable technique of clustering and cluster placement and leverages the tight integration between the algorithms by means of a common, unified data model.

1. Introduction

The complexity of today's largest IC designs is over ten million gates. Looking ahead it is prudent to prepare a methodology for the 50M gate ASIC. Three forces are at work which make designing chips at the edge of the capability of the fabrication technology increasingly difficult. First, the size of today's 10M gate designs taxes even the largest and fastest computers. Second, the deep sub micron (DSM) effects are breaking existing design flows. The third force is the shrinking market window. The capacity and complexity problems impact the productivity causing a product to miss its market window.

This leads to a requirement for a **fast, high capacity and scalable** technology that provides early estimates of post-layout performance and identifies many issues that would typically have been found only after detailed place & route in a conventional flow. This saves numerous time-consuming iterations and enhances the productivity to enable fast time-to-market. An additional advantage of such a technology is that engineers can explore design architectures and implementation alternatives and have a high level of confidence that performance goals can be achieved.

The approach being discussed in this paper is that of a silicon virtual prototype (SVP). A *silicon virtual prototype* is a fast physical implementation of the design. This implementation has been coarsely tuned to reduce some of the most time consuming analyses and optimizations. However, the silicon virtual prototype still has sufficient accuracy to identify long-wire type timing implementation issues that are prevalent in large designs. Virtual prototyping leverages its high capacity to have a global view of the chip design to identify problems and also automate the creation of a floor plan. While most of the steps can be done automatically, the opportunity for manual intervention exists in most intermediate stages. Thus, the process of quickly building a virtual physical prototype of the final design from RTL or netlist in order to estimate chip area, performance and power early in the design cycle and determine changes needed to the RTL or constraints, if any, in order to successfully achieve timing closure in implementation, is called silicon virtual prototyping.

The organization of the paper is as follows. In section 2 we review previous work in this area. Section 3 outlines the two goals of the methodology. Section 4 describes an overview of the methodology as well as the use model. Section 5 describes extensively and in detail all of the different technologies which are required in this methodology. Section 6 introduces the products that embody this methodology.

2. Previous work

This paper touches on many issues in many different areas of EDA, but we will only review previous work in some of the core areas of prototyping, estimation and clustering.

Early work in estimation [3][4][6][8] primarily focused on prediction of area, wire length and congestion. For instance [3] and [4] describe methods for forecasting the size and wire length of a design or partition. [6] and [8] use floorplanning methods to forecast congestion and routability. [5] and [11] describe approaches where floor planning information is used to support behavioral synthesis and can hence be seen as a form of prototyping.

Beyond area and routability, performance and timing are an important concern [1][10][13].The authors of [17] describe an algorithm which simultaneously addresses logic synthesis and placement. [16] describes a more comprehensive hierarchical floorplanning methodology addressing the performance and timing budgeting issues.

Prototyping technology raises the question of accuracy of fast estimation [18]. If the virtual prototype shows the design to be easily achievable then this approach provides a fast implementation flow.

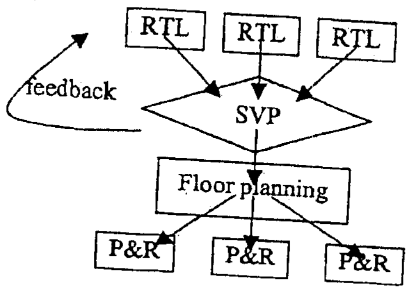
The methods that we use to develop a prototype are based on clustering. Clustering has been described in a number of placement and partitioning contexts. [2] and [7] describe the application of clustering in placement using simulated annealing algorithms, while [12][15] describe the application of clustering in force directed or quadratic placement optimization algorithms. [14] describes the application of multilevel clustering to graph partitioning.

0-7803-7659-5/03/$17.00 ©2003 IEEE.

3. Two Goals

Building a silicon virtual prototype (Figure 1) has two distinct goals. The first goal is to determine the feasibility of the design. The second goal is to develop a floor plan and constraints that can be used for the actual implementation of the design.

Figure 1: Building a Silicon Virtual Prototype



These joint goals are addressed by quickly creating a fast, flat implementation of the design. This flat implementation leverages the unique scalability of the placement problem. By creating clusters, the number of objects to place can be reduced dramatically. Making the clusters larger creates nearly unlimited speedup of the placement algorithm. This creates a continuous trade-off between run time and placement quality.

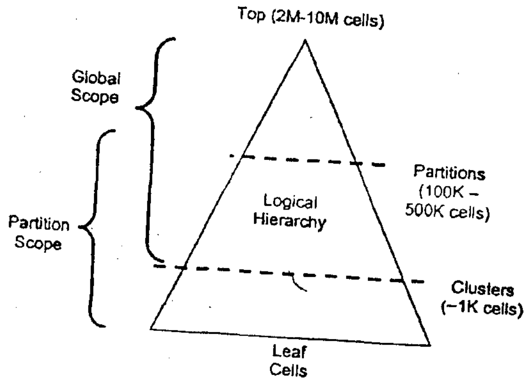
Feasibility concerns, especially early in the design cycle, are dominated by timing and performance issues. To do a feasibility check, it is essential that timing analysis is performed with reasonably accurate delays. Especially with today's multi-million gate chips, delay is very much affected by wire length. Long wires determine the performance of the design, and therefore cannot be ignored. Luckily, the cluster placement is sufficient to determine which wires are the long wires and to create a reasonably accurate wire length estimate for these wires. The short wires have much less of an impact on performance, and hence it is not necessary to complete a detailed placement to get accurate wire lengths for the short wires.

The second goal of virtual prototyping is to create a floor plan for the further hierarchical implementation of the design. The cluster placement can here be used as a guide to determine the appropriate partitions and to shape and position them. To drive implementation process for each of the partitions, it is necessary to set up the constraints in both the physical and the logical domains. In the physical domain, besides the shape of the partition, the most important constraints are the pin positions. In the logical domain, the relevant constraints are the timing constraints for the pins of the partition.

4. Proposed Prototyping Methodology

The prototype is a flat chip level design, in which global timing has been done. In order to speed up the placement, not actually each of the cells in the entire design is placed. Rather, the placement and optimization is done with clusters, which each consist of a number of cells (Figure 2). The number of clusters is several orders of magnitude smaller than the number of cells. The distances inside each cluster are small enough and the cluster sizes are more or less uniform, that the wiring does not play a role in the delay calculation.

Figure 2 : Tuning cluster size at different levels



Once timing closure has been achieved at the prototype level, the prototype can either be refined for final implementation or the design can be partitioned into *partitions*. Each of these partitions is then implemented flat. By making the clusters significantly smaller than the partitions, there is overlap of the global scope and the partition scope. Overlapping the global scope and the partition scope reduces the effects of the boundaries of the partitions on the quality of the placement.

The use model is one where a system integrator does daily integration. The timing closure loop consists of iterative prototyping runs while the RTL designer addresses timing issues from an architecture point of view. The methodology lets the system architect / system integrator learn about timing problems, by running the cluster placement and doing global timing. As timing problems are discovered, they can be solved at the "architectural" level. This can mean changing timing, pipelining, handshaking, FIFO queues, prefetching, speculative execution, parallel execution or by putting things physically close together. This is how the "architect" designs his chip. The end product of his/her design is the "golden RTL source".

To build a prototype, the system reads a Verilog or VHDL description, chip or partition level SDC timing constraints and any constraints for pad or macro placement. The system includes fast logic synthesis, static timing analysis, and placement. A daily prototype is built, starting with RTL and ending with global timing. Timing problems are identified as a result. The RTL designers work on the timing problems and try to resolve them. Now, in addition to all of the things a designer can do to speed up the design, the RTL can also be changed to impose physical proximity. By adding cluster placement constraint, the RTL designer has an additional tool to resolve timing closure problems at the architectural level. A design team will build several prototypes before committing to one particular partitioning.

The first step in the process consists of reading the Verilog source and clustering. In the case where the Verilog source is at the RTL level, this includes fast RTL synthesis. Once the net list has been read, the clusters in the hierarchy are identified. Ideally, most clusters already exist as modules in the logic hierarchy. Clusters should not be too large or too small. A global placement of the clusters is performed, which is fast, because there are only a few thousand clusters. Global static timing analysis can be performed on the placed clusters. This shows critical paths due to long wires. The cells inside the cluster are positioned in the center of the cluster. The global timing can also be used to derive timing budgets.

Once the prototype is complete, the design can be partitioned, a floor plan can be developed. Usually, partitions are chosen along the lines of the logical hierarchy, due to constraints imposed by the functional verification and test methodologies. Floorplanning involves giving each of the blocks a shape and a position. Both abutment style as well as channel style floor plans are supported. Once the floor plan has been determined, the timing constraints and the pin positions can be derived. This information is required for the separate implementation of each of the partitions.

5. Technologies Required for Prototyping

a. *Fast, high quality RTL synthesis technology*

One of the two goals of SVP is to support RTL design decisions and to arrive at full chip RTL timing closure. To do this, it is very important to perform accurate timing analysis and delay calculation. SVP provides much more accurate timing analysis as a global placement allows the identification of the long wires. The other requirement for SVP is to have very high capacity. It depends on synthesis technology which can synthesize several million gates at once— a 10x increase in capacity over conventional synthesis solutions. This means that, unlike conventional flows in which the RTL has to be arbitrarily partitioned into numerous submodules, our methodology allows design engineers to create their RTL based on functional intent rather than synthesis capacity limitations. Synthesizing the entire chip – rather than numerous submodules – produces superior optimization results because the algorithms are not limited by arbitrary boundaries.

The large ASIC designs are typically composed of multiple functional partitions that are in different stages of definition and implementation during the design process. If a SVP tool has to provide the early insight into the design issues it has to read in the design net list that would be a mixture of gates, RTL and black boxes (e.g. Hard IP). The SVP tool requires a synthesis engine, which supports black box models and can start prototyping from incomplete RTL.

As conventional synthesis tools are too time-consuming for SVP, some prototyping tools use "fast and dirty" synthesis like direct RTL mapping. This compromises the correlation between the results obtained from such a technology and the quality synthesis engine that will be used for the implementation. Poor correlation minimizes the value of the early feedback and not only makes the flow unpredictable but also does not prevent iterations between logic and layout.

b. *Clustering Technology*

The proposed virtual prototyping methodology is based on the unique scalability of placement by means of clustering, which is an abstraction technique for placement. It reduces the number of objects that need to be considered in an intuitive way: To partition a design of 10M cells into 10 paritions, it should not be necessary to consider the individual standard cells. The clustering factor gives the designer an easy way to trade off accuracy versus run time, which is important when cluster placement is used to evaluate floor-planning decisions.

Clustering is primarily driven by the connectivity of the net list. Placement clusters should be small enough to be uninteresting to the designer. A clustering algorithm should generate them automatically without user interaction.

The objective is to select clusters that have few connections to other clusters. The actual number of cells vary from cluster to cluster to keep the areas of the clusters as uniform as possible. Having uniform cluster areas is good for placement, because placement algorithms are better at handling uniformly sized objects. If the clusters are too large then the deviation between the cluster placement used for the optimization and the final detailed cell placement can nullify the accuracy advantage. Similarly if the clusters are too small then the runtime gets adversely impacted.

Since the clusters are small, the shape of the cluster is not very important, hence it has been chosen to be a square with an area proportional to the area of the cells contained in the cluster. For the purpose of placement, all of the cells contained in a cluster are assumed to be at the center of the cluster. The exception is very large cells, such as macro's, which are the only cell within a cluster. Such clusters have the area and shape of the macro itself.

Clusters follow the logical hierarchy, meaning that they only group cells that have the same parent in the hierarchy. If the logical hierarchy is deep and there are modules of all sizes in the logical hierarchy, then it is often advantageous to use the modules of the logical hierarchy as clusters. Since these have names that are meaningful to the RTL designer, the resulting placement is much easier to read for the RTL designer. Sometimes timing constraints are attached to the boundaries of the modules, and therefore their preservation is more important. The RTL designer is already used to using the logical hierarchy as a tool to influence the implementation of the design. Hence, keeping a logical module together is in line with the intentions of the RTL designer, even if it is not optimal with respect to wire length. Keeping elements in logical hierarchy together gives the RTL designer the opportunity to make physical proximity decisions as architectural decisions.

The clusters form a separate hierarchical data structure, which represents the physical hierarchy. This second hierarchy coexists simultaneously with the logical hierarchy. The clusters are allowed to contain a hierarchical cell, which is interpreted as if they contain every leaf cell in the hierarchy tree rooted by the hierarchical cell. In this manner the physical and logical hierarchy may be interwoven. The physical hierarchy has a second, higher level, the level of the partition. The partition is a grouping of clusters in the physical hierarchy.

c. Cluster Placement Technology

The clusters are placed using a force directed placer, which uses a second order non-linear optimization algorithm. While clustering reduces the number of placeable objects, the number of nets decreases sub-linearly. After clustering, there are many nets that connect exactly the same set of clusters. A single net with a higher weight can replace these nets.

After the placement is first performed, a timing verification can be performed. This is the main feasibility check on the implementability of a very large design. The placement of the clusters shows where the long wires are. This information is used in the delay calculation of the long wires.

d. Fast optimization technology

Fast optimization technology is used to implement the logic with the required accuracy. For combinational logic, the speed advantage is obtained through synthesis methods based on the concept of logical effort. The concept of logical effort has a simple and elegant relevance – the delay depends on the gain of the gate, and not on its exact parasitics. The resulting breakthrough is that the ratio of capacitances can be chosen within limits beforehand, and can be kept constant by gate sizing during the design implementation stages. By adopting this methodology, delay can be controlled without advanced knowledge of parasitic capacitances. More importantly, circuit evaluations are performed without guessing at or fixing cell sizes before the actual routing exists.

As the design progresses the gain of each gate is carefully tuned, delays are spread over paths, and the timing is maintained. This methodology has a unique advantage that the gain-based approach results in slew equalization where every cell in the design is sized such that it has exactly the drive strength for that path to meet timing. This is done for all paths whether they are critical or not and due to this approach SI issues are greatly reduced as there are no unnecessarily overdriven or under driven nets which would have played the role of aggressor or victim respectively.

All of this means that there are a number of very important outcomes from the use of gain-based synthesis:

- Synthesis times are dramatically reduced compared to traditional synthesis techniques. In one typical real-world example, a multi-day evaluation with a traditional synthesis tool was reduced to less than 5 hours with gain-based synthesis tool (Blast Create).
- The relative simplicity of Blast Create's gain-based calculations means that it has a far greater capacity than other synthesis solutions. In turn this means that Blast Create has the capacity to handle multimillion-gate designs without resorting to artificial partitioning.

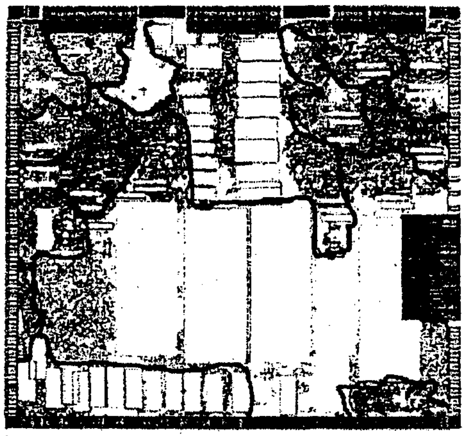
The end of the synthesis step completes all timing optimizations, and all of the circuit delays are determined and frozen.

e. Partitioning

The partitioning methodology works as follows: When the clusters are first placed, the modules in the net list hierarchy form amorphous areas. The system includes graphical means to visualize the placement of the various levels in the logical hierarchy (Figure 3). This forms a guide to the user to determine the choice and final shape and location of top-level partitions of the hierarchical floorplan. Figure 1 shows clusters in a full chip. Each cluster is represented by a different color (shown here by marking the borders explicitly).

The user will have to decide on the appropriate partitions based on the placement of the clusters. Modules in the logical hierarchy are candidates for partitions. Hierarchy manipulation can be used to arrive at good partitions. A heuristic will assign shapes and positions to each of the partitions. The user can intervene at this point and shape the partitions and give them a position.

Figure 3: Cluster representation



Since the partitions are usually rectangular, the clusters will need to be placed into these rectangles. This is accomplished in a second cluster placement stage. In this stage each of the partitions is represented by a region constraint or "fence", which forces the clusters to remain within their assigned regions.

f. Time budgeting

For the front end RTL designer, the most important aspect of the prototype is timing verification and timing budgeting. From timing verification the RTL designer desires an accurate forecast to determine if timing closure of the RTL design can be achieved. If not, the RTL designer needs feedback as to what measures can be taken to improve the timing of the design. An RTL designer works on a design to improve the timing by changing the architecture of the design.

Timing budgeting uses gain trimming approach which adjusts the delays of the cells to get zero slack at all points in the design. Gain trimming works as follows: a factor (the gain) is adjusted on outputs of cells depending on the slack. Outputs with a positive slack are made slower and outputs with a negative slack are made faster, proportionally to the slack and the number of levels of the path. Since there are a lot of reconvergent paths, which influence each other, this has to be repeated to converge to zero slack. This requires multiple timing analysis runs. Once zero slack has been achieved at all points, the timing constraints can be derived by characterization. As the required and arrival times are the same, either one, or any combination can be used to form the timing constraints if the design will be implemented hierarchically or it can be incrementally refined for final flat implementation.

g. Pin assignment

To implement each of the partitions, it is necessary to determine the pin positions on the boundary of the partitions. This must be done carefully, taking into account the requirements of antenna rules, power grid, as well as congestion. The pin assignment depends directly on the cluster placement. Because clusters are smaller than partitions, and clusters have been placed within the partitions, they give detailed placement information of the cells within each of the partitions. This information is used in pin assignment by routing from cluster to cluster.

Two methods for pin assignment are available. The first method is a simple heuristic based on point to point connections. This method is suitable for interactive work, as pins can be assigned one at a time, or by group. The second method is based on global routing. This method is more sophisticated, as it takes congestion into account.

h. Testability analysis technology

The RTL design can have structural testability issues that would require an iteration to modify the source. The structural issues can be identified early with net list checking tools. The SVP identifies the implementation issues. As the repair for testability issues modifies the netlist this also has to be embedded in the flow that produces the final optimized netlist. As we have stated before that logic synthesis is an integral part of the SVP flow, we need to include DFT analysis and repair as a key required technology.

Failure by logic designers to adequately apply scan design rules early in the design cycle can lead to poor test coverage and testability problems later in the product development cycle. Hence DFT checks including checking for gated clock, bi-directional and tristate bus contention, set/reset inputs help identify and fix, if needed, such problems early in the design cycle.

i. Correlation to implementation technology

A key requirement for a SVP is that it correlates with the actual implementation results. The SVP tools in the market today face a challenge that building interfaces between disparate tools has put most of the RTL-to-GDSII flows together. These typically have multiple analyses engines like extractor and timers and have correlation issues within the flow so it is impossible for an external SVP tool to not have inaccuracy issues. Moreover, the use of different placement and global routing algorithms can lead to a substantial miscorrelation with the actual implementation.

High performance or high-density designs require accuracy of optimization. Clock tree, power structure and scan contribute significantly to the use of routing resources. Also the impact of placement quality cannot be neglected. Poor placement increases overall wire length, and reduces performance, and worse, it leads to invalid assumptions for optimization. Moreover,

reducing optimization in the early stages of the design methodology can lead to larger increases later in the flow.
Magma has the unique advantage that it has common analyses & optimization engines in its integrated flow from prototyping to final implementation. Magma also has a unified data model enabling seamless sharing of all data throughout the complete flow. This ensures that most of the algorithms and all of the analysis engines are shared, guaranteeing close correlation between the prototype and the actual implementation.

J. Glass box Technology

Assembling a design, which has been created using a hierarchical methodology, requires a strategy to reduce the amount of data. The entire design consists of the partitions plus the routes to connect the partitions. In order to assemble the design, it is necessary to have certain data for each of the partitions; in particular, it is necessary to have the pin positions and IO timing, but it is not sufficient. For accurate delay calculation, it is necessary to have a detailed delay model for the input and output wires of each of the partition inputs and outputs. To do antenna rule fixing, a detailed model for each of the wires is required.
One approach is to develop model for each of the many different applications that require analysis of the partitions: delay calculation and antenna rule check are just two of many types of analysis. Other types of analysis are noise analysis, clock skew analysis, detailed routing modeling, design rule checking and timing analysis. This however, requires special data representation, often with special data formats, to be developed for each model, together with the data abstraction methods. This constitutes a considerable code development, test and maintenance burden.
The other approach is not to require the development of special purpose models, but to inspect the design of each of the partitions itself. This has the advantage that the actual design data is inspected in the same manner as the design data is inspected during flat design. After extraction, the actual routing geometries become the "model" for delay calculation, while the same geometries can be inspected for antenna rule violations. While this is considerably easier to implement, the amount of data for each of the partitions is prohibitive.
The solution is to reduce the actual design data, by selective deletion of features, which are not relevant for the chip assembly. In particular, for timing purposes, cells that are contained between register boundaries, with no combinational path to a partition IO, cannot influence the top level timing. Similarly, for antenna rule checking, it is only necessary to inspect the routing geometries between an IO pin and the cells within the partition. Preserving these geometries allows the entire route to be checked using the same antenna rule checking code that is used for a flat design. Other routes can be deleted, assuming that they have been implemented correctly at the partition level.
We have developed data reduction techniques along these lines. The data reduction techniques delete all design elements (cells, nets, routing geometries) which cannot affect any global analysis. While there are many different global analysis methods to take into account, experience shows that in most cases, a large amount of data can be removed. The amount of reduction tends to grow with the size of the partition, so this methodology will scale very well with future technology generations.
Since there is a single, comprehensive file format to represent all design data, no new format for abstraction needs to be invented. Moreover, this single format can support all present and future analysis requirements. No separate formats or "views" are required to support different analysis requirements.

6. Product Description

Blast Create allows logic designers to check, visualize, evaluate and improve the quality of the RTL code and design constraints. A logic-level prototyping or structural analysis is useful in the detection of large muxes, snaking timing paths, fanout violations, combinational loops or deep logic levels. Early detection of such problems leads to good quality RTL, hence a good quality netlist that finally leads to a better implementation. This facilitates a clean ASIC handoff model. The checking is enhanced by visualizations of the RTL functionality that lets designers view schematic representation of their code, view registers and latches, logic-clouds as well as allows them to directly cross-probe into the source RTL. Blast Create also facilitates testability checks via *Blast DFT* that is seamlessly integrated within Blast Create.
Blast Create delivers the only viable prototyping solution for today's complex designs with fast, high-capacity synthesis and fully integrated analysis and implementation engines that operate on a common datamodel. Figure 4 shows a chart of experimental data using Blast RTL on various designs. The chart is shown for the size of design vs runtime in hours. Table 1 shows some of the experimental results of Blast Create usage on real designs.

Figure 4: Experimental results from Blast RTL

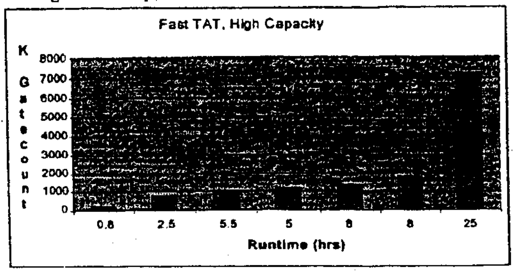


TABLE 1: Experimental results from Blast Create

	A	B	C
Design Size (Gates)	1.3M	6.0M	23.2M
Macros	104	69	–
Cluster Size	200	10	1000
Cluster Placement	6 min	2.5 hrs	2 hrs
Prototype	2.8 hrs	7 hrs	20 hrs
Memory	1.5 GB	317 MB	10GB
Platform	32bit Linux	32bit Linux	64bit Solaris

It is important that the prototype correlates well with the actual implementation tool i.e. the quality of results should not be compromised. Blast Create is implemented on the same code base as *Blast Fusion*; it uses the same timing analysis and shares the same high quality placement and optimization engines.

Blast Create allows you to rapidly place the entire design to create a prototype of the chip. It allows you to place the design including the macros. Floor planning constraints and manual decisions can be quickly evaluated by iterating the placement interactively. To speed up the placement, clusters of cells are placed.

Blast Plan is the portion of the product which uses SVP to create a floor plan. This tool is targeted for the back end designer, who, faced with a very large design, needs to create a floorplan for the hierarchical implementation of a design. Blast Plan allows the selection and creation of partitions, and most importantly, it allows the creation timing constraints and pin positions for the separate implementation of the partitions. Blast Plan also contains the abstraction technology for the bottom up hierarchical assembly of a large chip.

7. Conclusions

We have presented a methodology for the design of next generation of multi-million gate chips. The methodology is based on the construction of a "Silicon Virtual Prototype", which allows both an early timing feasibility check as well as the construction of a globally optimized floor plan.

8. References

[1] Modarres, H. and S. Raam, J.-H. Lai: "Floorplanning of hierarchical layout in ASIC environment" in: Proc. IEEE Custom Integrated Circuits Conf., sec. 7.1, Rochtester, May 1988.

[2] Mallela, S. and L.K. Grover: "Clustering based simulated annealing for standard cell placement" in: Proc. 25th ACM/IEEE Design Automation Conf. pp.312-317, Ahaheim, June 1988.

[3] Chen, X. and M.L. Bushnell: "A module area estimator for VLSI layout", in *Proc. 25th ACM/IEEE Design Automation Conf.* pp.54-59, Anaheim, June 1988.

[4] Pedram, M. and B. Preas: "Accurate prediction of physical design characteristics for random logic", in: Proc. IEEE Int. Conf. Computer Design, pp.100-108, Oct. 1989.

[5] McFarland, M.C.: "A fast floor planning algorithm for architectural evaluation" in *Proc. IEEE Int. Conf. Computer Design*, pp. 96-99, Cambridge, Oct. 1989.

[6] Lengauer, T. and R. Muller: "A robust framework for hierarchical floorplanning with integrated global wiring", in: Proc. Int. Conf. on Computer-Aided Design, pp.148-151, 1990.

[7] Lee, Y.W. and Y.S. Cheung, C.S.K. Yeung: "A flexible clustering and floor planning approach to standard cell placement using hierarchical simulated annealing", in: *Proc. Int. Conf. Circuits and Systems*, pp.882-885, Shenzhen, 1991.

[8] Pedram, M. and E. Kuh: "BEAR-FP: A robust framework for floorplanning", in: Int. J. High-Speed Electron., vol.3, no.1, pp.137-170, 1992.

[9] Ding, C.L. and C.-Y. Ho, M.J. Irwin: "A new optimization driven clustering algorithm for large circuits" in Proc. Eur. Design Automation Conference, pp.28-32, Hamburg, Sept. 1993.

[10] V. Narayananan and D. LaPotin, R. Gupta, G. Vijayan: "PEPPER – a timing driven early floorplanner", in Proc. IEEE Int. Conf. Computer Design. pp.230-235, Austin, 1995.

[11] Mecha, H. and M. Fernandez, F. Tirade, J. Septien, D. Motes, K. Olcoz: "A method for area estimation of data path in high level synthesis", in IEEE Trans. Computer Aided Design, Vol 15, no. 2, pp. 258-265, Feb. 1996.

[12] Chen Chunhong, and Tang Pushan: "Cluster-based placement for macrocell gate arrays", in Proc. 2nd Int. Conf. on ASIC, pp.46-49, Shanghai, Oct. 1996.

[13] Bushroe, R. G. and S. DasGupta, A. Dengi, P. Fisher, S. Grout, G. Ledenbach, N. S. Nagaraj, R. Steele: "Chip hierarchical design system (CHDS): A foundation for timing-driven physical design into the 21st century", in: Proc. Int. Symp. Physical Design, pp.212-217, 1997.

[14] Alpert. C.J. and Jen-Hsin Huang, A.B. Kahng: "Multilevel circuit partitioning" in: IEEE Trans. Computer Aided Design, vol.17, no. 8, pp.655-667, Aug. 1998.

[15] Xianlong Hong and Hong Yu, Changge Qiao, Yiel Cai: "CASH: a novel quadratic placement algorithm for very large standard cell layout design based on clustering", Proc. 5th Int. Conf. Solid-State and I.C. Technology, Beijing Oct. 1998.

[16] Su, Hsiao-Pin and A.C.-H. Wu, Y.-L. Lin: "A timing-driven soft-macro placement and resynthesis method in interaction with chip floor planning", in Proc. 36th Design Automation Conf. pp.262-267, New Orleans, June 1999; Also in: IEEE Trans. Computer Aided Design, Vol 18, no. 4, pp.475-483, April 1999.

[17] Salek, A.H. and Jinan Lou, M. Pedram: "An integrated logical and physical design flow for deep submicron circuits". IEEE Trans. On Computer Aided Design, Vol. 18, no. 9, pp.1305-1315, Sept. 1999.

[18] Sarrafzadeh, M. and M. Wang: "Can fast algorithms be used as good predictors?", in: Syst. Level Interconnect Prediction, pp. 125-1999.

[19] Ranjan. A. and K. Bazargan, M. Sarrafzadeh: "Fast hierarchical floorplanning with congestion and timing control", in: Proc. 2000 Int. Conf. Computer Design, pp.357-362, Austin, Sept. 2000.

[20] Ranjan A. and K. Bazargan, S. Ogrenci, M. Sarrafzadeh: "Fast floorplanning for effective prediction and construction", in. IEEE Trans. VLSI Systems, Vol. 9, no. 2, pp.341-351, April 2001.